Data Express 4.0

Getting Started with Distributed Data Stores

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
http://www.microfocus.com

2019-02-08

Contents

Getting Started with Distributed Data Stores

This guide contains explanatory information and a series of tutorials that show you how to start using Data Express. The tutorials introduce the major features of Data Express in a number of short sessions, typically 10 - 30 minutes.

This guide is intended for customers who have the Data Express for Distributed Systems solution or for customers who have the Data Express for z/OS solution but have also purchased the additional ODBC Extension or Oracle Extension.

Note: Previous to the 4.0 release, the Data Express tool suite was intended for users who analyze and manage data stores from System z servers running z/OS. Now with the advent of Data Express 4.0, the suite of Data Express tools is available to non-mainframe users who can now analyze and manage data stores from Windows. In addition, Data Express 4.0 now provides support for distributed data stores for z/OS and Windows users, via the ODBC Extension and Oracle Extension.

Using Distributed Data Stores with Data Express

Data Express 4.0 introduces the ability to mask and subset ODBC-enabled data stores. This new functionality basically mimics the same approach in methodology and functionality provided with the Oracle Extension Technology in previous version of Data Express.

ODBC-enabled data stores are supported on both the z/OS and distributed platforms in the same way.

Note: The Oracle Extension workflow is the same as the ODBC Extension workflow with one exception: the input of multiple data stores for Data Masking and Data Subset Extraction is not supported.

Usernames and passwords are case sensitive and null passwords are allowed when connecting to a database.

The work-flow process for ODBC-enabled data stores looks like this:

- Data Inventory Phase (Data Builder)
  - Load from Database (Distributed Loader)
  - Load from File (Distributed Loader)
- Data Classification Phase (Data Builder)
  - Manual Association
  - Import Classification from Referential Integrity
  - Import Classification from Data Dictionary
  - Import Classification from Sampling Results
  - Limited Life Cycle Support
    - Skip, Overwrite, Update (Distributed Loader)
  - Data Analysis
    - Sampling
    - Fingerprinting
- Data Masking Phase (Data Masking)
  - Metadata preparation

    - Export masking information (Distributed Exporter)
    - Actual masking (Extension Technology)
- Data Subsetting Phase (Data Subset Extraction)
    - Method preparation
        - Create Referential Integrity methods
    - Export method information (Distributed Exporter)
    - Actual subsetting (Extension Technology)
    - Distributed Statistics Loader to enable run-time statistics

DBA Tasks for Extension Technology

You can update your Run-Time Knowledge Base to enable Data Express extension technology. Depending on the current state of your Run-Time Knowledge Base, you can make the updates by executing one or more SQL scripts as specified in the topics listed below.

Script Execution

With regard to the execution of SQL scripts to make the updates, the following applies:

- All scripts must be executed by a database user with DBA privileges.
- By default, all scripts provided by Micro Focus grant privileges to PUBLIC. You can change this to limit privileges solely to the user ID(s) used to connect to your source data store(s) as dictated by your database security requirements.

User Credentials

For each database user to mask and subset data using the ODBC or Oracle Extension, you must grant access to source and target data stores as well as the Run-Time Knowledge Base, located in the source data store(s). The privileges required are:

Object	**Required Privileges**
Source data store(s)	SELECT on tables processed within the specified schema(s)
Run-Time Knowledge Base	INSERT, SELECT, and DELETE on tables within the DEKB schema
Target data store(s)	CREATE TABLE, INSERT, and DELETE on tables processed within the specified schema(s)

New Installations

Before using either the ODBC or Oracle Extension, a Run-Time Knowledge Base needs to be created within your source data store(s). The Run-Time Knowledge Base drives the operation and performance of Data Express' masking and subsetting processes. The Run-Time Knowledge Base consists of tables and indexes, and needs to be created using the schema name DEKB.

For convenience, we provide scripts that are tailored for working with data stores. To use these scripts, download the `3441.scripts.zip` file from the *DBA tasks for working with the Data Express Extension Technology* article located in the Micro Focus Knowledge Base. This zip file contains individual scripts for the following data stores:

- IBM DB2 LUW
- Microsoft SQL Server
- Oracle

If you are using a data store for which no specific script is provided, use the `createkb.sql` script, located by default in your Data Express installation directory, as the basis for the creation of the Run-Time

Knowledge Base. This script contains "canonical" DDL (Data Definition Language) statements, describing the Run-Time Knowledge Base tables and indexes.

Before executing a script file:

IBM DB2 LUW
Edit the script file and update its CONNECT statement and connection credentials to connect to the appropriate database.

Microsoft SQL Server
Edit the script file and update its USE statement to specify the database in which to create and populate the Run-Time Knowledge Base.

Oracle
Create a database user named DEKB and grant the DEKB user all privileges required to create and populate the Run-Time Knowledge Base.

`creatdb.sql`
To ensure that `createkb.sql` executes correctly for all source data stores edit the file and, if necessary, update the CREATE TABLE and CREATE INDEX statements such that they enable the successful creation of the DEKB schema, and have the appropriate authority to create and populate that schema.

Data Express Upgrades

When upgrading your installation from Data Express 4.0 or earlier, you must update the HSURDFIL table definition in all instances of the Run-Time Knowledge Base.

For convenience, we provide scripts that are tailored to update the HSURDFIL table definition in each Run-Time Knowledge Base. To use these scripts, download the `2045.AlterKB.zip` file from the *DBA tasks for working with the Data Express Extension Technology* article located in the Micro Focus Knowledge Base. This zip file contains individual scripts for the following data stores:

- IBM DB2 LUW
- Microsoft SQL Server
- Oracle

Required Run-Time Knowledge Base Updates

Before installing any Data Express update, you must update your run-time tables by executing the following for each Run-Time Knowledge Base that uses the IBM DB2 LUW, Microsoft SQL Server, or Oracle data stores:

```
CREATE TABLE DEKB.HSLOGTAB(MSGCODE CHAR(7), FLAGENVI CHAR(1), ROUTNAME
CHAR(8), VALRETCODE CHAR(2));
GRANT SELECT, UPDATE, DELETE, INSERT ON DEKB.HSLOGTAB TO PUBLIC;
```

For more information, see *Logging Exit Routines* in the *Data Masking Guide*.

If you are using a data store other than BM DB2 LUW, Microsoft SQL Server, or Oracle, update your run-time tables as required to allow updates to table definitions and grant the permissions required to update the DEKB schema.

Centralized Run-Time Knowledge Base for the ODBC Extension

Note: This topic applies to the ODBC Extension only.

Because Data Express extension technology requires that the Run-Time Knowledge Base contains tables and indexes created using the DEKB schema, the DEKB schema must be in the source data store. If this requirement compromises production integrity when using the ODBC Extension, use a centralized Run-Time Knowledge Base.

To configure a centralized Run-Time Knowledge Base, you must create a new configuration file named `hkblayer.ini` and store it in the executables directory of the ODBC Extension installation (see *Extension Technology Directories* for more information). This configuration file defines the connection information required to access a centralized Run-Time Knowledge Base.

Include the following content in the `hkblayer.ini` file:

```
RDBMS = ODBC
DBNAME = ConnectionAlias
DBOWNER = TableOwner
USER = UserID
PSWD = Password
```

ConnectionAlias	The data store containing the Run-Time Knowledge Base tables. For the ODBC Extension, this is the data source name (DSN).
TableOwner	The table qualifier for the Run-Time Knowledge Base tables.
UserID* and *Password	The connection credentials required by the data store containing the Run-Time Knowledge Base tables.

For example:

```
RDBMS = ODBC
DBNAME = MYDSN
DBOWNER = MYOWNER
USER = MYUSER
PSWD = MYPASSWORD
```

After creating the `hkblayer.ini` file, start a command prompt and change to the executables directory that contains it. Enter the following command to generate the `HKBLAYER.rc` file required to access the centralized Run-Time Knowledge Base:

```
dxeencrypt hkblayer
```

Using Distributed Loader

The Distributed Loader is accessible in Data Builder from ▤.

For distributed data stores, the Distributed Loader is a utility that lets you load information from an ODBC-enabled database to the Knowledge Base as part of the Data Inventory process. The Knowledge Base can exist either on Windows or z/OS.

Loading Data Store Information

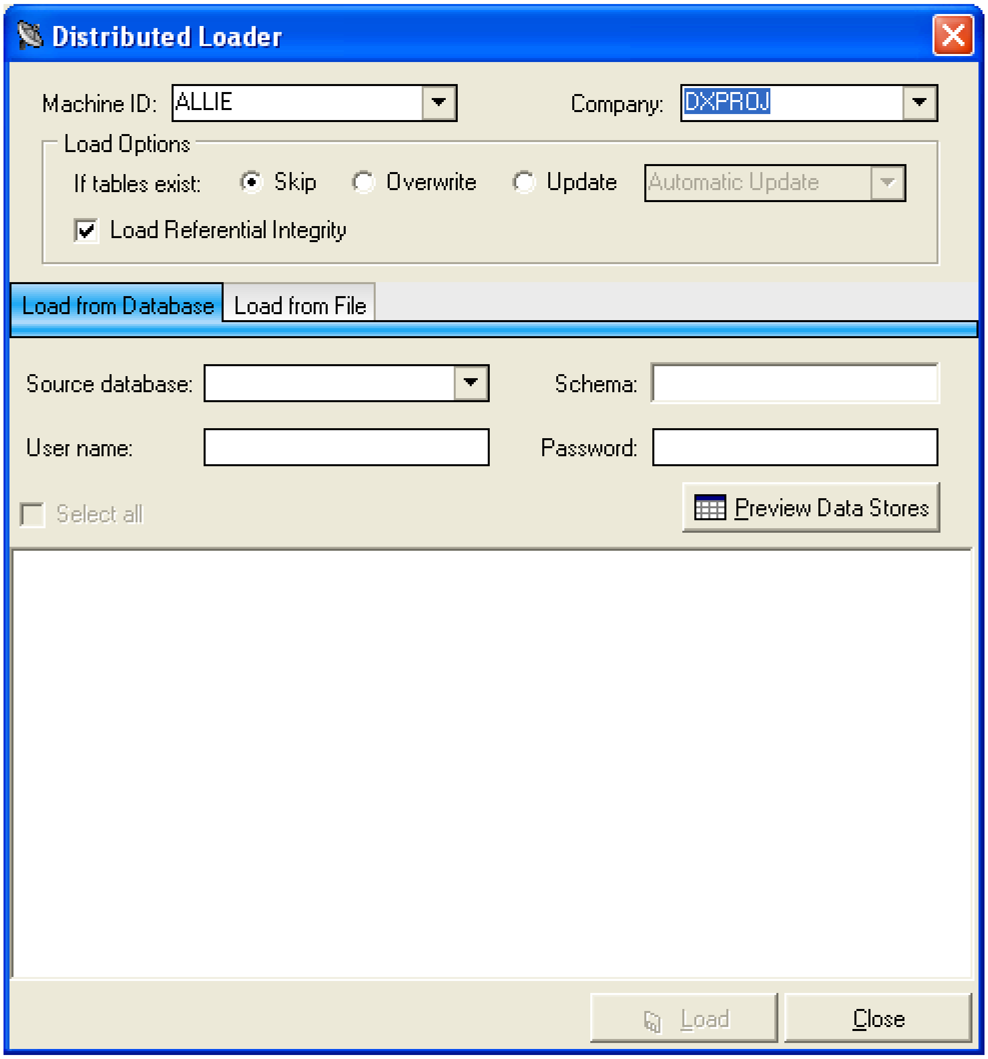
Metadata information about your data stores can be loaded into Data Express in one of two ways: from a database or from flat files.

Not only can the Distributed Loader load information about your tables, columns, and indexes, but it can also work with the referential integrity information contained in your distributed data store.

Load from Database

When you choose to load information from a database, the Distributed Loader provides a list of all the ODBC-enabled data stores defined in your Windows environment, and lets you select the data store's metadata information you want to load into the Data Express Knowledge Base.

To load information about your data store from a database, begin by selecting the **Load from Database** tab.



Distributed Loader

Machine ID: ALLIE

Company: DXPROJ

Load Options

If tables exist: ● Skip ○ Overwrite ○ Update Automatic Update

☑ Load Referential Integrity

Source database:

Schema:

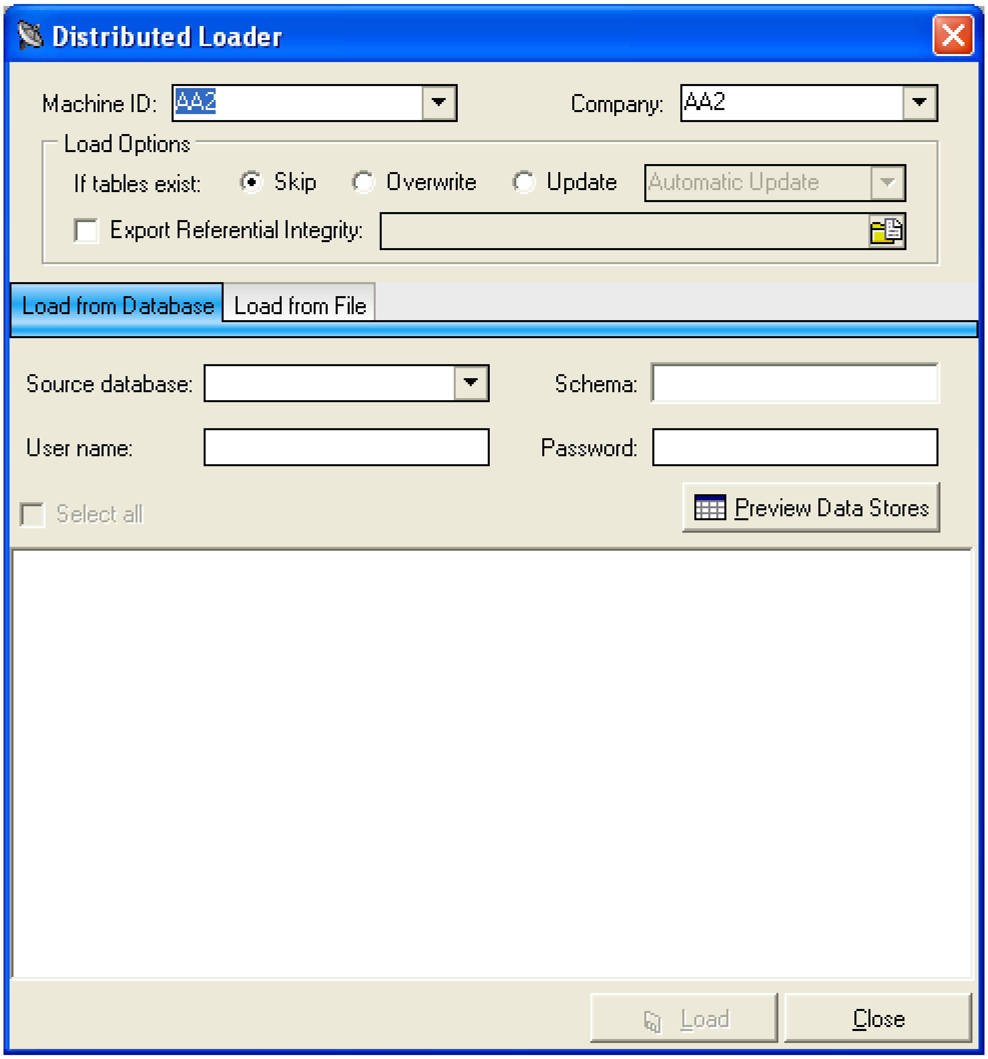
User name:

Password:

□ Select all

In Data Express for Distributed Systems, you will see an import checkbox for referential integrity labelled: **Load Referential Integrity**. This provides the ability to load your referential integrity information into the Windows Knowledge Base, in addition to standard schema information about your tables. You are now ready to import your referential integrity classes. See *Importing Classes* for more information.

The Distributed Loader utility differs slightly for the z/OS solution:


Distributed Loader

Machine ID: AA2 Company: AA2

Load Options

If tables exist: ● Skip ○ Overwrite ○ Update Automatic Update

□ Export Referential Integrity:

Source database: Schema:

User name: Password:

□ Select all

In Data Express for z/OS, an export checkbox labelled **Export Referential Integrity** is provided instead. In this environment, a file that contains referential information is created for you. Once this file is transferred to the mainframe, you can load and import your referential integrity classes. See *Importing Classes* for more information.

The ability to export referential integrity is based on an administrator's permission. This permission is provided by updating the ACTIVATION field in the HSURDRIA table in the knowledge base. By default, the table is empty. In order to allow the usage of referential integrity, the administrator needs to insert a value 'Y' in that field and to provide privileges to the user connecting to it.

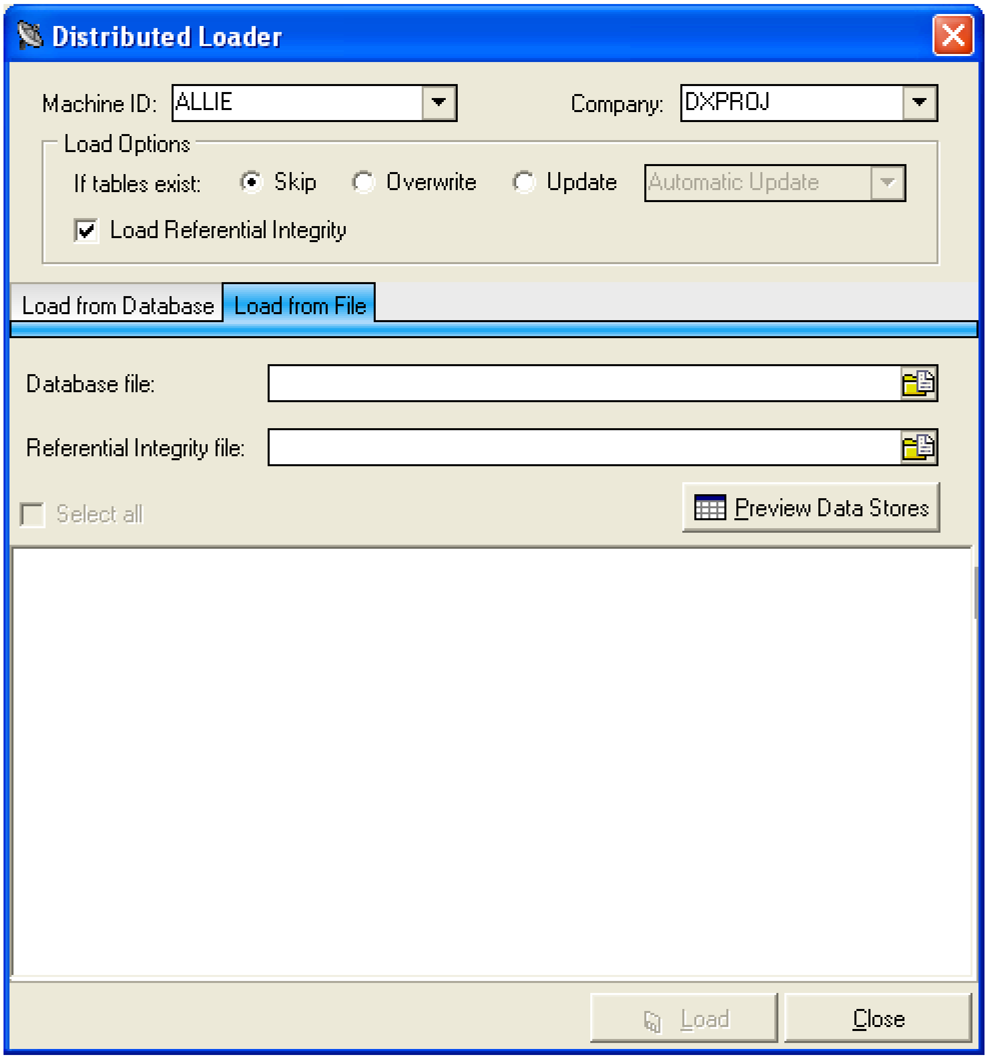
Standard schema information about your distributed data store's tables is automatically loaded in the z/OS knowledge base. You can logically reduce the amount of information you wish to load by specifying the name of a schema in the **Schema** field. Otherwise, you leave this field blank and the Distributed Loader retrieves and loads all the information about all the schemas in the data store.

For more information about the **Load from Database** feature, review the **Data Express for Distributed Data Stores Tutorials** in this guide.

Load from File

Sometimes you may not be able to access your distributed data store from Windows. For this situation, the Distributed Loader provides the ability to load data store information from flat files.

To load information about your data store from a file, begin by selecting the **Load from File** tab. In the Distributed Systems solution, the **Load from File** tab of the Distributed Loader appears similar to the following:



Machine ID: ALLIE

Company: DXPROJ

Load Options

If tables exist: ● Skip ○ Overwrite ○ Update Automatic Update

☑ Load Referential Integrity

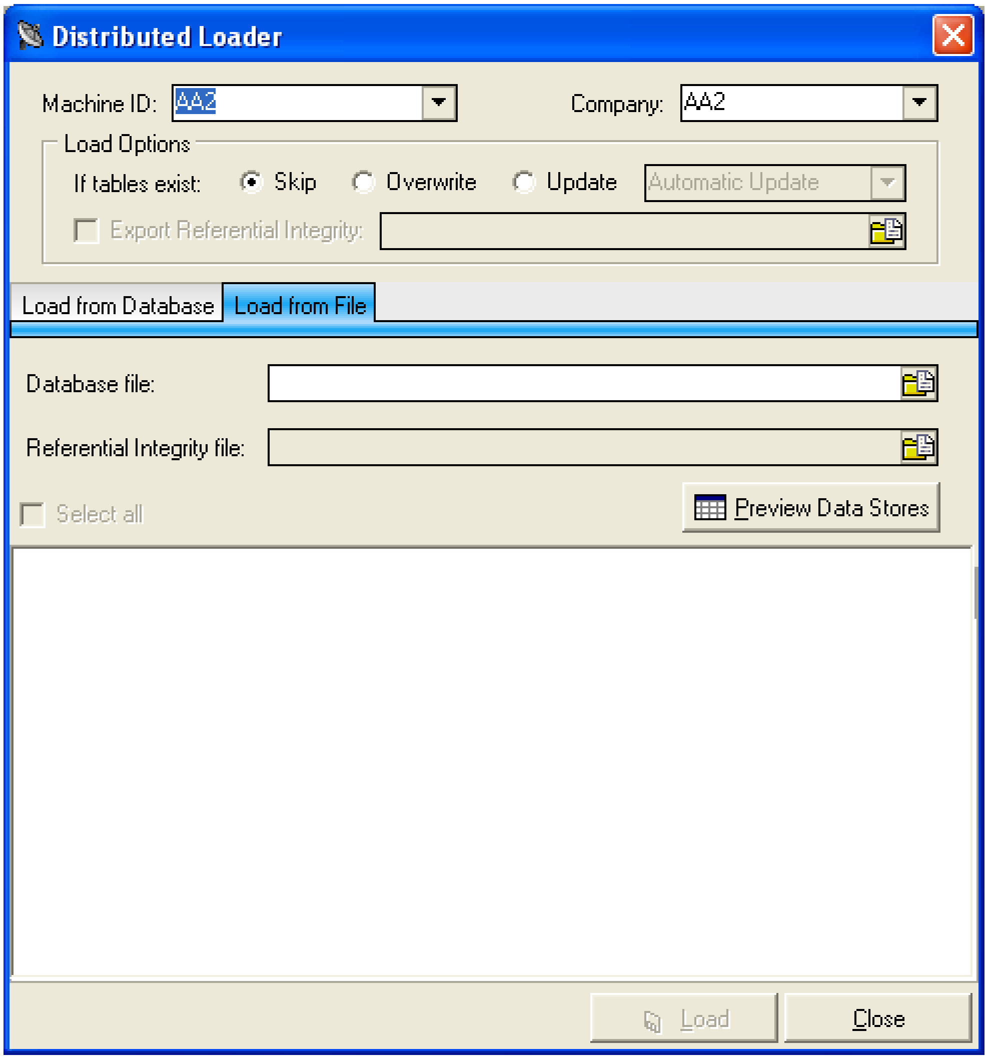
Database file:

Referential Integrity file:

□ Select all

In Data Express for Distributed Systems, two files can be used as input to the load process. The first file contains information about your tables, columns, and indexes. The second optional file contains information about your data store's referential integrity.

The **Load from File** tab of the Distributed Loader is different for the z/OS solution:



Machine ID: AA2

Company: AA2

If tables exist: ● Skip ○ Overwrite ○ Update   Automatic Update

□ Export Referential Integrity:

Database file:

Referential Integrity file:

□ Select all

In Data Express for z/OS, only the file that contains information about your tables, columns, and indexes can be used. For a z/OS Knowledge Base, your referential integrity information gets loaded into the Knowledge Base while executing the **Import Classification from Referential Integrity** job. See *Importing Classes* for more information.

The files used as input by the Distributed Loader are created by the Extension Technology utility: **dxegenloadfile**. This utility creates both files.

If you are using Data Express for z/OS, the referential integrity file created by **dxegenloadfile** is used as input to the **Import Classification from Referential Integrity** job.

If you are using the ODBC Extension or the Oracle Extension and want Data Express to load the user ID and password for the selected connection, run the **dxegenloadfile** using the **-c** parameter. This generates an input file with table information in the corresponding ODBC or Oracle log directory. See *Extension Technology Directories* for more information.

For more information on **dxegenloadfile**, see *Extension Technology Utilities* in *Using Extension Technology*.

Limited Life Cycle Support

For distributed data stores, both in z/OS and Distributed environments, Data Express provides limited Life Cycle support.

After you have loaded information about your data stores in Data Express, your table definitions may have changed. For this possibility, the Distributed Loader provides the following options:

- **Skip** - if the data store to be loaded already exists in the Knowledge Base, previously loaded tables are ignored.
- **Overwrite** - if the data store to be loaded already exists in the Knowledge Base, old definitions are removed and the new definitions are inserted. If the data store is new, definitions are added. This option clears all the data element-class assignment in the existing data store.
- **Update** - if the definition for the data store exists in the Knowledge Base, the following occurs:
  - Metadata is added to the Knowledge Base for new data elements in the data store.
  - Metadata for data elements that have different size values (length or precision or scale) are updated in the Knowledge Base, but information for assigned classes remains untouched.
  - Metadata for data elements that have different data types are updated in the Knowledge Base and all class assignments are cleared.

Use of Data Analysis (Sampling) for Class Assignment

Data Express for distributed data (both in the configuration with MVS knowledge base and in the configuration with XDB knowledge base) allows the capability of analyzing data content by using the sampling feature.

This feature helps in understanding the data element content and in assigning it to the appropriate class. The sampling process provides two types of results:

- **Standard sampling** : it is a list of values with corresponding occurrences of the data elements selected for the process. This result can easily be checked via the user interface by human intervention.
- **Compressed sampling** : this is a statistical distribution of the values of the data elements selected for the process, represented in a grid with the first character and number of characters of each value contained in the data element. The result is designed to be used by an algorithm based on class assignment by analogy (data elements having the same class are supposed to have the same statistical distribution of values).

The result of the sampling can be used to make class assignment in two different ways:

- Manual assignment can be done by checking the standard sampling result and then by assigning classes to data elements through the "Work with data elements" feature. Note that the user interface is the same one of Data Express for MVS
- Using assignment from sampling result. In this case, compressed sampling results are used. This process is further composed of two separate parts:
  - The requirement is defining one or more prototypes for each specific class, this happens in the "Work with data elements" feature by dragging a "Selected data element attribute" to a "Class sample". That is associating to a class the compressed sampling of a data element containing the same class.
  - Then an interactive process starts, and this process assigns the class to all data elements included in the parameters of the interactive process itself, basing the class assignment on the similarity of the compressed sampling results. This process compares the compressed sampling of the class to the compressed sampling of the data elements included in the parameters of the process itself, and if they fit it, assigns the class to the corresponding data element. Note that Data Express for MVS uses a similar but different one, based on the submission of a batch job from "Work with jobs" feature.

The sampling process will be described more in detail in the "Using Distributed Sampling" chapter, and it will be object of a tutorial described in the present manual.

The standard sampling and compressed sampling result verification is described in details in the "Work with data store" chapter of the Front End User Guide.

The manual class assignment and the prototype definition are described in details in the "Work with data elements" chapter of the Front End User Guide.

The assignment from sampling result is described in details in the present manual, in paragraph "Sampling result" of "Importing classes" chapter.

Importing Classes

After you have completed the Data Inventory phase using the Distributed Loader, you need to import this information so that Data Express can create the corresponding classes. This is called the class generating process and is part of the Data Classification phase.

To begin the class generating process:

1. From Data Builder, navigate to the **Work with Data Elements** window (click ▤ or access it through **Environment** > **Work with Data Elements**).
2. Click the **Classes - Assignments** tab.
3. Click **Import Class**. The **Import Class** window is displayed.

Data Express offers three options at this point: you can either create classes for referential integrity **(Referential Integrity**), supply your own definitions of new classes for the columns in your tables (**Data Dictionary**), or you can create classes for related data elements after sampling.

Note: All options are available to the Distributed Systems solution and the z/OS solution when accessing distributed data stores. However, if you are using the z/OS solution for non-distributed (mainframe) data stores, sampling is handled by the **Class Data Element Assignment** and **Data Store Data Element Sampling** jobs. For more information about sampling with non-distributed z/OS data stores, see section *Data Analysis* in the ***Process Guide for z/OS*** .

Data Dictionary

The **Data Dictionary** option gives you the ability to define your own classes for your distributed data stores from a flat file format.

For the format of the Data Dictionary input file, see section *Import Classification from Data Dictionary* of the ***Data Model Guide*** .

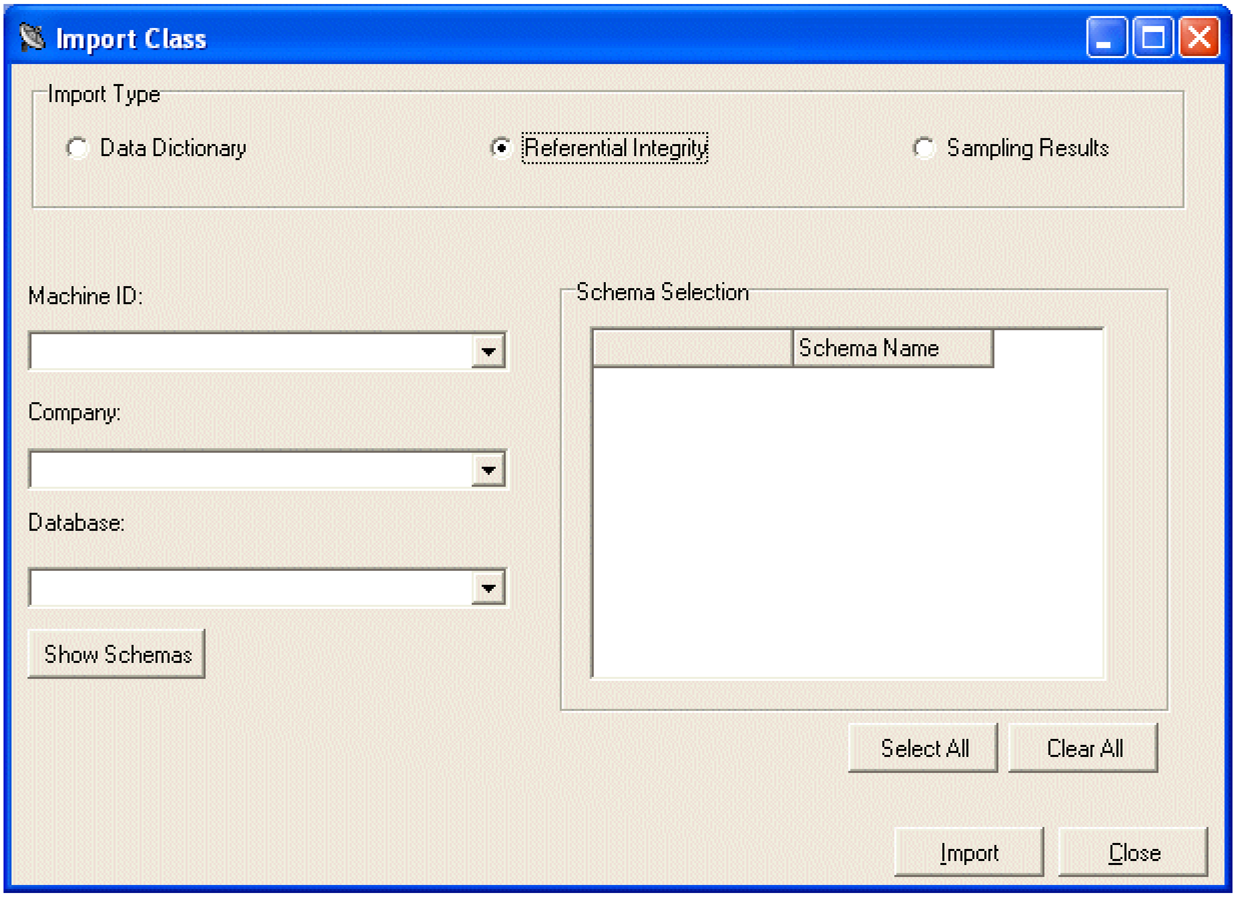
For both the Distributed Systems and z/OS solutions, the **Import Class** window is the similar when **Data Dictionary** is selected.

For the **Data Dictionary** Import Type in the Distributed Systems solution, you simply need to provide the location (data store) of the Windows text file that contains your new class declarations, in addition to declaring the appropriate workspace (machine ID, company name). The classes are generated once you click **Import**.

For the **Data Dictionary** Import Type in the z/OS solution, you simply need to provide the location of the text file that contains your new class declarations (date store), in addition to declaring the appropriate workspace (machine ID, company name). Remember that this is a z/OS file specification. Select **Import** to schedule the **Import Classification from Data Dictionary Job**.

Referential Integrity

For both the Distributed Systems and z/OS solutions, the **Import Class** window is similar when **Referential Integrity** is selected:



Import Class

Import Type

○ Data Dictionary ● Referential Integrity ○ Sampling Results

For the **Referential Integrity** import type, you need to provide your workspace information (machine ID and company), distributed data store (database), and schema specification. The classes are automatically generated once you click **Import**. Select **Referential Integrity** if you want to create referential integrity classes.

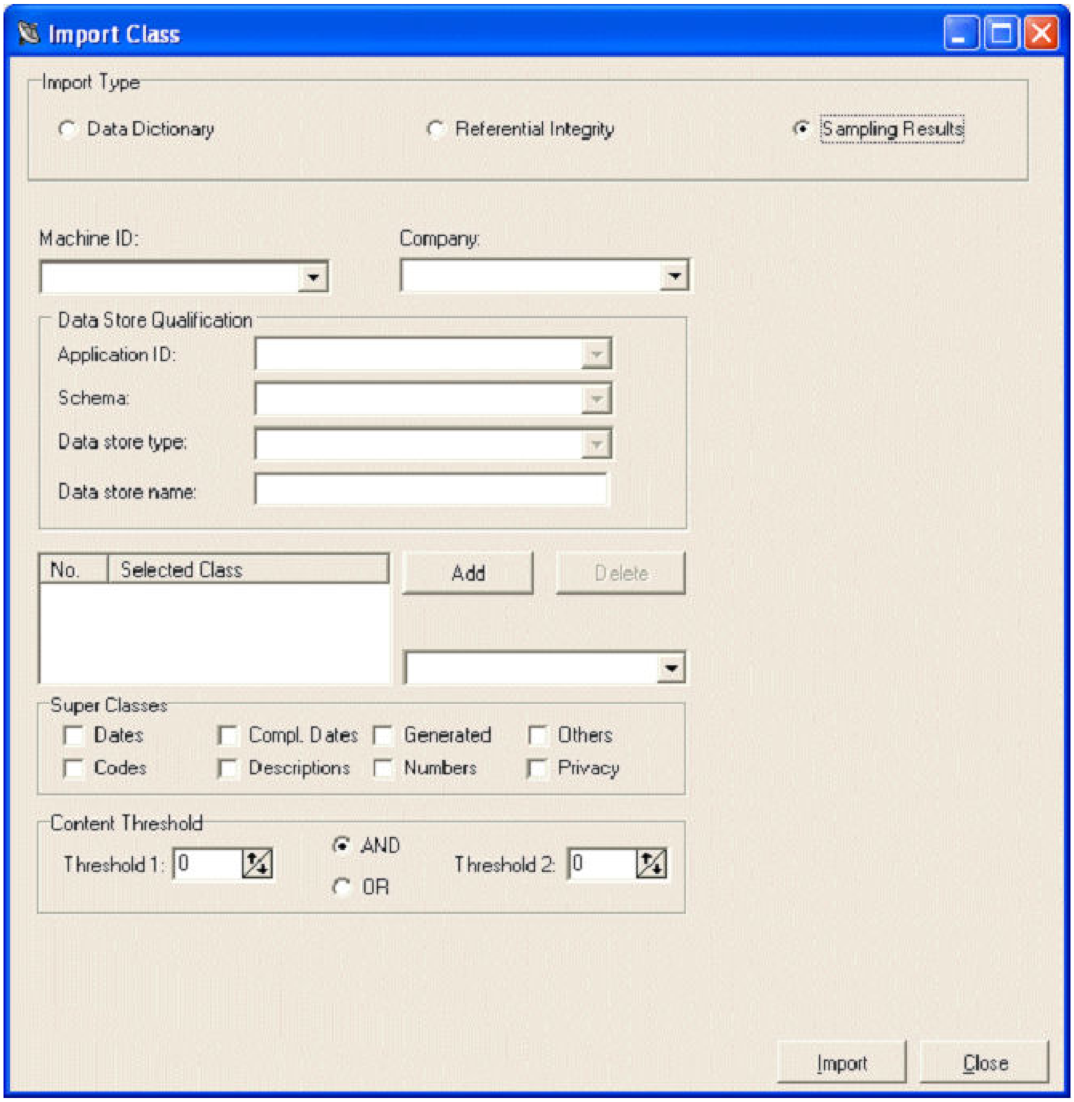
For the **Referential Integrity** Import Type, you need to provide your workspace information (machine ID and company), distributed data store name (database), and additional fields provided in Data Store Qualification. Select **Import** to schedule the **Import Classification from Referential Integrity** job.

More information about the import classification jobs can be found in the ***Process for z/OS Guide*** , the ***Data Model Guide*** , and the ***Front End Guide*** .

Sampling Results

The **Sampling Results** option gives you the ability to import your class assignments after you have assigned a fingerprint to your class. The class assignments are based on confidence values calculated using threshold recognition algorithms to determine the similarity of contents.

For this **Import Type**, you need to provide your workspace information (machine ID and company) and **Content Threshold** values. If desired, you can select other data store information to further restrict the class assignments. Select **Import** to assign classes to the data elements that have a fingerprint that matches the prototype fingerprint to the degree specified by the **Content Threshold** values.



Import Type: ○ Data Dictionary ○ Referential Integrity ● Sampling Results

Threshold 1: 0 ● AND ○ OR Threshold 2: 0

For both the Distributed Systems and z/OS solutions, the **Import Class** window is the similar when **Sampling Results** is selected. However, it is only appropriate to import your classes in this way if your data stores are distributed (it works both with DB2 on MVS knowledge base and with XDB knowledge base).

Here is a summary of the parameters used by this function:

- **Machine ID -** Identifier of the machine containing the data stores where Import Class process will work.
- **Company -** Identifier of the company containing the data stores where Import Class process will work.
- **Data Store Qualification -** With this set of selection parameters, it is possible to specify the scope of the class assignment process (which data stores you want to assign to the specified class).
- **Selected class -** By using the **Add** botton it is possible to select one or more classes for the assignment process. This mean the process will attempt the assignation of the selected class or classes.
- **Super Classes -** Selects a super class. This mean the process will attempt the assignment of the selected superclass.
- **Thresholds -** Logical operations between **Treshold1** and **Treshold2**. Data Express calculates two distinct formulas in order to understand how the processed column data distribution fits with the one of the prototype (by comparing compressed sampling result). Each one of the two formulas returns a number in the range 0-100, that is as higher as the two distributions are similar. By setting the two thresholds, and by setting the and/or parameter, it is possible to guide Data Express in the class assignment based on comparison between prototype and column to be assigned.

If you are working with z/OS data stores and a mainframe KB, you must schedule a series of jobs to perform the sampling functionality for you. For instance, you would schedule the job **Data Store Data Element Sampling** to select the sampling configuration, schedule the job **Class Data Element Assignment** to suggest class assignments, and then confirm the class assignments in the **Work with Data Elements** window of Data Builder.

For more information about assigning classes for your z/OS data stores, see section *Data Analysis* of chapter *Project Implementation* in the ***Process for z/OS Guide***

For more information about importing classes by sampling results for distributed data stores, see the *Simple Sampling* exercise in this guide.

Creating Referential Integrity Methods

After you have imported your referential integrity classes, you can create subsetting methods that adhere to those specifications.

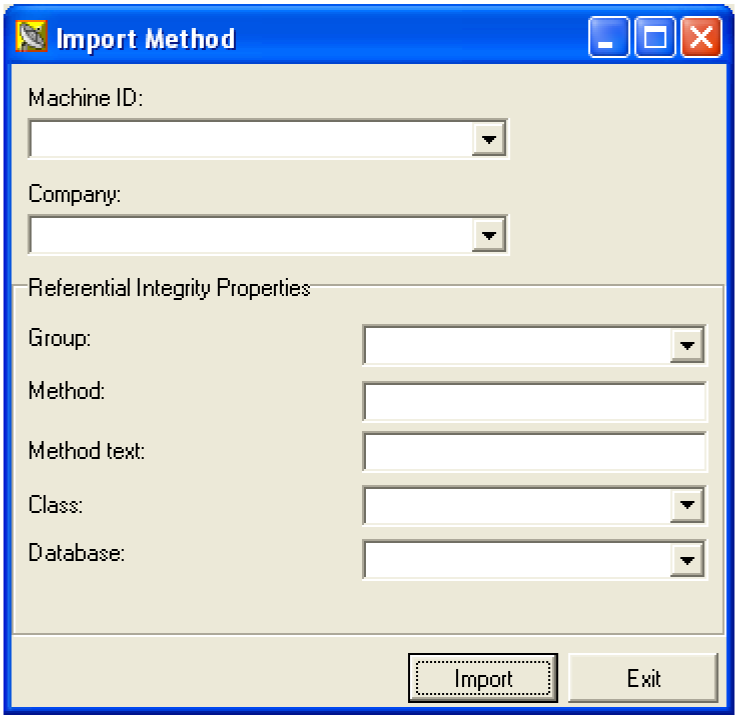
Data Express for z/OS

In Data Express for z/OS, you will need to schedule the **Import Method from Referential Integrity Job**. See the ***Process for z/OS Guide*** and the ***Front End Guide*** for more details.

Data Express for Distributed Systems

In Data Express for Distributed Systems, from Data Subset Extraction, you can create a Referential Integrity method automatically.

Click (Import Method from Referential Integrity) to access the **Import Method** window:



After supplying your workspace and method group, you need to supply the new method's name and description. In addition, you must also supply the class that drives the referential integrity (the leading class), and the distributed data store.

After selecting **Import**, the Referential Integrity method is created.

For more information, review the *Data Express for Distributed Data Stores Tutorials* in this guide.

Using Distributed Exporter

The Distributed Exporter utility is accessible from both Data Masking and Data Subset Extraction.

Its role is reasonably straight-forward; it creates and provides necessary information to the Extension Technology so that actual masking and subsetting of your distributed data store can occur.

There are two areas of interest when it comes to using the Distributed Exporter with distributed data stores:

- If you use ODBC-enabled data stores, you can mask and subset across all your data stores with one invocation of the ODBC Extension. This feature is not available with the Oracle Extension.
- You also have the ability to subset and mask within one distributed data store, as long as a distinct target schema name is provided.

For both Data Masking and Data Subset Extraction, the information you need to provide are very similar. For both scenarios you need to provide:

- your workspace (machine ID and company)
- your source and target distributed data store credentials
- directory where the output files for the Extension Technology should be placed

For Data Subset Extraction, you can only export one method at a time by design.

Data Masking

To access the Distributed Exporter, click from the main window of Data Masking.

To export from Data Masking, you need to provide the following input:

- Your workspace (machine ID, company)
- Your source data store credentials (source database, source schema, source user ID/password)

  Note: The source database is an ODBC DSN configured to access that data store.
- Your target data store credentials (target database, target scheme, target user ID/password)

  Note: The target database is an ODBC DSN configured to access that data store. The Extension technology does not support case sensitive object names.
- Directory where you want to place output files for Extension Technology (Output directory)

Note: If the source database and the target database are the same, the schemas cannot be the same.

In addition, if you are using ODBC-enabled data stores, you can provide multiple sets of source and target data store credentials.

After you have successfully exported from Data masking, the following files are created in your output directory:

- **method.rc** - Encoded connection criteria for each data store referenced.
- **elab.txt** - List of data stores and the enabled columns within those data stores that need to be masked.
- **CREATETABLE.sql** - list of the tables that need to be masked; a generic CREATE TABLE statement is provided for each table.
- **CREATEINDEX.sql** - list of all the indexes for the tables that need to be masked; a generic CREATE INDEX statement is provided for each index.
- **ALTERTABLE_RI.sql** - list of all the primary and foreign keys for the tables that need to be masked; a generic ALTER TABLE statement is provided for appropriate primary and foreign keys.

Note: SQL files can be used while utilizing the Extension Technology; see section *SQL Generated by Distributed Exporter* for more details.

Data Subset Extraction

To access the Distributed Exporter, click from the main window of Data Subset Extraction.

To export a method from Data Subset Extraction, you need to provide the following input:

- Your workspace (machine ID, company)
- Your method (group, method)
- Your source data store credentials (source database, source schema, source user ID/password)

  Note: The source database is an ODBC DSN configured to access that data store.
- Your target data store credentials (target database, target schema, target user ID/password)

  Note: The target database is an ODBC DSN configured to access that data store. The Extension technology does not support case sensitive object names.
- Directory where you want to place output files for Extension Technology (Output directory)

Note: If the source database and the target database are the same, the schemas cannot be the same.

In addition, if you are using ODBC-enabled data stores, you can provide multiple sets of source and target data store credentials.

After you have successfully exported from Data Subset Extraction, the following files are created in your output directory:

- **method.rc** - Encoded connection criteria for each data store referenced.
- **method.txt** - Information identifying the method.
- **filter.txt** - Information about the filters used in the method.
- **elab.txt** - Information about the elaborations in the method.
- **cbfld.txt** - Information about combined data elements used in the method.
- **CREATETABLE.sql** - list of the tables that need to be masked; a generic CREATE TABLE statement is provided for each table.
- **CREATEINDEX.sql** - list of all the indexes for the tables that need to be masked; a generic CREATE INDEX statement is provided for each index.
- **ALTERTABLE_RI.sql** - list of all the primary and foreign keys for the tables that need to be masked; a generic ALTER TABLE statement is provided for appropriate primary and foreign keys.

Note: SQL files can be used while utilizing the Extension Technology; see section *SQL Generated by Distributed Exporter* for more details.

SQL Files Generated by Distributed Exporter

Whenever you use the Distributed Exporter to export masking or subsetting information for consumption by the Extension Technology, three SQL files are created. These three files (**CREATETABLE.sql**, **CREATEINDEX.sql,** and **ALTERTABLE_RI.sql**) contain "canonical" DDL (Data Definition Language) statements about the source distributed data store's tables, indexes, and primary/foreign keys.

Normally, the Extension Technology will create tables in the target distributed data store with the same table specifications as in the source; however, no indexes or referential integrity constraints are created. If the Extension Technology "sees" that the tables already exist, it simply places the masked or subsetted data into those already created tables.

If you want your target distributed data store to more closely mimic the schema of your source distributed data store, these SQL files can help you in that effort.

Note: The SQL files are not guaranteed to execute properly with all ODBC data stores. Each ODBC data store can have slightly different syntax when it comes to CREATE TABLE, CREATE INDEX, and ALTER TABLE statements.

- Because of this, Micro Focus recommends the following before you attempt to execute the statements:
  - Review the syntax of the SQL before you execute it to ensure it works with your distributed data store.
  - Before executing the SQL, make sure you have **userid** credentials sufficient to execute DDL statements in your distributed data store.

If you successfully execute the contents of these three files in your target distributed data store before using the Extension Technology, the Extension Technology will mask or subset data accordingly.

Using Distributed Sampler

The Distributed Sampler is accessible from Data Builder from ▦.

For distributed data stores, the Distributed Sampler is a utility that lets you perform sampling on your data stores and then load that information in the Knowledge Base as part of the Data Inventory process.

Sampling lets you obtain information about the distribution of your data. You can use the distribution as a way to assign classes to other data elements with similar data distributions.

There are three types of sampling:

- **Compressed sampling** - Produces the data element *fingerprint*. The fingerprint graphically shows the distribution of values within a given range for the sampled data element. The fingerprints for numeric and alphanumeric data elements differ in that the fingerprint for an alphanumeric data element shows the distribution of values based on the first character and provides additional information.
- **Standard sampling** - Displays information for each data element value including the number of times each value occurred and the percentage that value represents in the total population.
- **Min/Max calculation** - Displays the minimum and maximum values for the data element.

Known Restriction: Data Express does not support sampling for binary data.

For more information about the **Sampling Options** feature, review the *Simple Sampling* Tutorial in this guide.

Sampling Options

Basic Options

Use the **Sampling Options** tab in the Distributed Sampler to specify your sampling criteria.

To export your sampling configuration, you need to provide the following information:

- your workspace (machine ID, company)
- your source data store credentials (source database, source schema, source user ID/password)

  Note: The source database is an ODBC DSN or Oracle TNS alias configured to access that data store.
- types of sampling to be performed
- directory where you want to place output files for Extension Technology (Output directory)

  Note: Only specify an **Output directory** if you are exporting your sampling configuration.

To export your configuration, click **Export**. After you have successfully exported from the Distributed Sampler, the following files are created in the specified output directory:

- **method.rc** - Encoded connection criteria for each data store referenced.

Note: If you have performed export functions for masking or subset extraction, the **method.rc** file already exists in your **config** directory. If you export your sampling configuration to the **config** directory, the previous **method.rc** file will be overwritten.

- **sampling.dat** - Information regarding the data elements to be sampled.

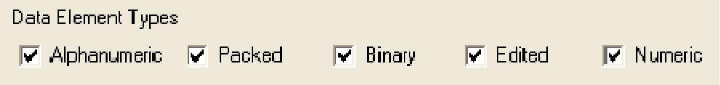
Note: If your Knowledge Base and the data stores you want to sample are located on the same Windows machine that is running Data Express, you do not need to export configuration information, execute sampling from a script, or load results. Instead, you would bypass those steps and click the **Start** button in the Distributed Sampler to execute sampling.

Advanced Options

To further control your sampling results, you can specify advanced sampling options, which are accessible from the **Advanced** button.

Additional Data Element Attributes

You can restrict which data elements are sampled by their data element types:



Data Element Types

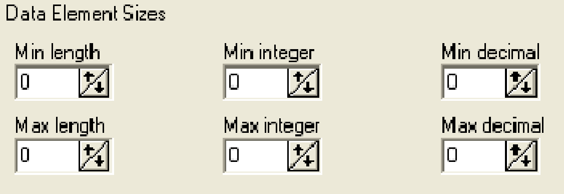
☑ Alphanumeric ☑ Packed ☑ Binary ☑ Edited ☑ Numeric

By default, all types are checked. Uncheck a type to remove it as a sampling candidate.

Note: The **Binary** data element type represents the COBOL binary (COMP field) data type. True binary is not supported.

Data Element Sizes

You can restrict which data elements are sampled by their data element sizes:



Data Element Sizes

Min length: 0
Min integer: 0
Min decimal: 0
Max length: 0
Max integer: 0
Max decimal: 0

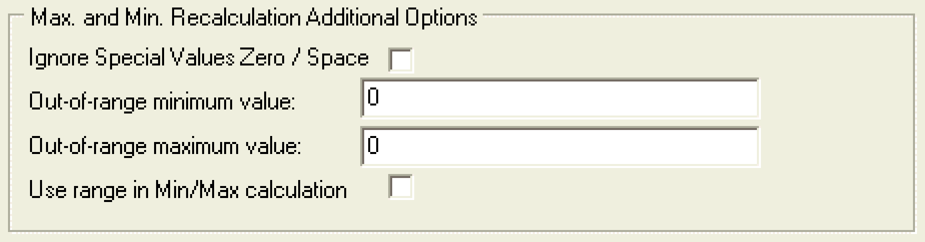
By default, all values are **0** (zero), which means to include all data elements regardless of the associated size values.

If a minimum value is set, the data element size to be included is the value greater than the specified *MinValue.* For example, if the **Min length** is changed to **1**, data elements with a length of 2 or greater are included.

Likewise, if a maximum value is set, the data element size to be included is the value less than the specified *MaxValue*. For example, if the **Max length** is changed to **99**, data elements with a length of 98 or less are included.

Max. and Min. Recalculation Additional Options

You can restrict the Min/Max calculation sampling for data elements:



Max. and Min. Recalculation Additional Options

Ignore Special Values Zero / Space □
Out-of-range minimum value: 0
Out-of-range maximum value: 0
Use range in Min/Max calculation □

The following options are available:

- **Ignore Special Values Zero / Space** - If this box is checked, the 0 (zero) and special characters like the space character are ignored in your minimum and maximum value calculations.
- **Out-of-range minimum value** - If a defined range within your data element values is desired, the **Out-of-range minimum value** field defines the smallest value outside of that range. For example, if **5** is specified, values of **6** and greater are included in the range.

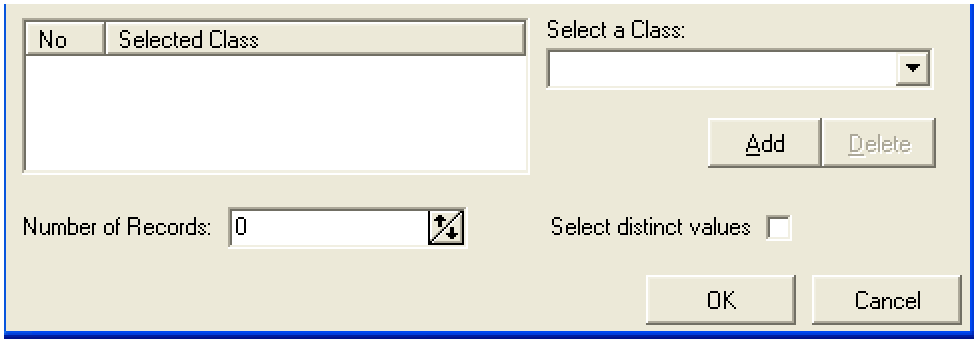
  By default, the **Out-of-range minimum value** is **0** (zero), which means to include all data elements regardless of the actual value.
- **Out-of-range maximum value** - If a defined range within your data element values is desired, the **Out-of-range maximum value** field defines the largest value outside of that range.

  By default, the **Out-of-range maximum value** is **0** (zero), which means to include all data elements regardless of the actual value.
- **Use range in Min/Max calculation** - If this box is checked, the range indicated by the fields **Out-of-range minimum value** and **Out-of-range maximum value** is applied to the Min/Max calculation sampling result.

Known Restriction: The **Max. and Min. Recalculation Additional Options** currently affect the minimum and maximum data element values reported for standard and compressed sampling.

Additional Restriction Options

You can restrict which data elements are sampled based on data element class assignments. You can also reduce the number of data elements to sample from.



No	Selected Class

Number of Records: 0

Select distinct values □

You can add a class to the table show on the left after selecting it in the **Select a Class** list. Only data elements that are assigned to a class specified in the Selected Class list will be sampled. This is especially helpful when you have several data stores containing data elements with multiple class assignments.

The **Number of records** is the total number of records to be sampled. The default value is **0**, which means to include all records.

Use the **Select distinct values** check box to improve the sampling process for big tables. The *SELECT DISTINCT* command is used to provide the distinct values of a column. This option allows you to create a sampling result only for the distinct values of a column. When checked, already sampled values are discarded and the result will be set for unique ones.

Note: The list of data elements is sorted either numerically or alphanumerically as based on the data element type. Once the limit of records is reached, the remaining data element records are not sampled.

Validation

Distributed sampling allows two types of execution:

Direct Automatically exports comma-separated side files (`sampling.dat` and `method.rc`) in the `config` subfolder of the Data Express extension installation path. It produces sampling results, and then imports those results into the client knowledge base.

This type of sampling requires no validation.

Manual You can manually export the comma-separated side files, sample manually from the console, and then load your sampling results into the client knowledge base from the **Distributed Sampler** dialog box in Data Builder.

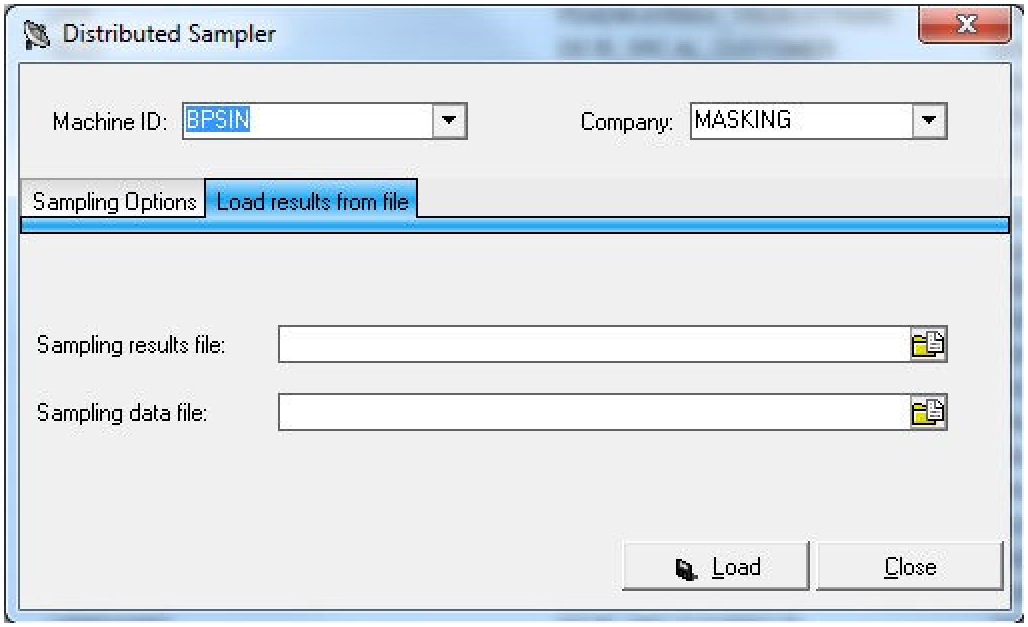
In this case, validation is useful to ensure that the data stores in the sampling results file match the exported `sampling.dat` file. This prevents errors and ensures consistent results.

Validation is performed when you specify a **Sampling data file** in the **Distributed Sampler** dialog box. See *Load results from file* for instructions.

Load results from file

During sampling execution, the Extension Technology uses the output files **sampling.dat** and **method.rc** to produce the **sampling.log** file that contains the resulting statistics.

The sampling analysis results contained in the **sampling.log** file must be loaded into the Knowledge Base from the **Load results from file** tab in the Distributed Sampler. To do this, click the browse button that corresponds to the **Sampling results file** field; then browse to and select your **sampling.log** file:



Machine ID: BPSIN

Company: MASKING

To validate the results, click the browse button that corresponds to the **Sampling data file** field; then browse to and select your **sampling.dat** file (optional).

Click **Load** to load the results.

Once you have loaded your results, you can view them in the **Show Synthetic Data Element Contents** window.

Using Extension Technology

This section of your Getting Started guide provides an overview of the Extension Technology, and provides detailed information on creating masking routines.

You should review the configuration information and the invoking procedure prior to starting the tutorials in *Data Express for Distributed Systems Tutorials*.

For information regarding hardware, software, user, and configuration requirements, see the *Distributed Systems* chapter in your ***Readme*** . In particular, see section *Run-time Knowledge Base Requirements* as you must create a Run-time Knowledge Base in order to mask or subset data.

For information regarding installation procedures, see the *Distributed Systems Procedures* chapter in your ***Installation Guide*** .

For more information pertaining to the execution of the ODBC Extension, follow the exercises in *Data Express for Distributed Systems Tutorials* of this guide.

Extension Technology Configuration

Configuration capabilities for the Extension Technology can be globally implemented or can be controlled at the data store level.

Extension Technology Configuration File

The `dxeconfig.cfg` file is used to globally manage your ODBC Extension or Oracle Extension settings. By default, it contains the following settings:

```
* Data Express Configuration File

LogLevel:1
Always Drop Table:N
CSV Data Store:N
COBOL Masking:N
Validate Target Table:N
Validate Source Schema:N
Enable Logging:N
Hide Progress:N
Hide Log Message:N
Use Extended C Masking:N
Trim Spaces for ODBC Char Columns:N
Commit Size(MBs) [user range = from 1 to 4000, default range = 0]:0
Parse String finding LOW-VALUES:N
Create Table using CHAR clause:N
Execute scripts when the process aborts:N
Execute disable/enable CONSTRAINTS:N
Execute Aborted Tables:N

* These entries are only applicable to the Oracle Extension.

Text Output:N
Data Separator:,
Fields Enclosed By:"
Identify Externally:N
Use Append Condition:N
```

The following settings can be specified:

LogLevel

Controls the level of detail displayed in the `dxe_errorlog.txt` file. You can change the **LogLevel** as needed or when requested to do so by Product Support:

LogLevel	Description
1	Default log level that provides a summary for each processed data store.

LogLevel	Description
2	Verbose output used for diagnostic purposes that contains the SQL statements executed against your source and target data stores.
3	Generates ODBC trace output that is written to `odbctrace.txt`.

Always Drop Table

Determines whether or not to always execute DROP TABLE, and what, if any, additional action is taken. Possible values are:

N	If a table already exists within the data store, the Extension Technology first executes a **DELETE FROM *TableName*** statement before performing **INSERT** operations. Default.
Y	If a table already exists within the data store, it is removed by the execution of the **DROP TABLE *TableName*** statement, which forces the table to be recreated.
K	Keep. When the target table is not already deleted, the existing records in the target are kept, and a new record with the same key as another record in the target is discarded.
L	Delete and keep. When the target table is not already deleted, it is deleted with the same query used to extract the record from source, then existing records in the target are kept, and a new record with the same key as another record in the target is discarded.
O	Overwrite. When the target table is not already deleted, the existing records in the target are kept, and a new record with the same key as another record in the target overwrites that record, in effect deleting the original record.
P	Delete and overwrite. When the target table is not already deleted, it is deleted with the same query used to extract the record from source, the existing records in the target are kept, and a new record with the same key as another record in the target overwrites that record, in effect deleting the original record.

Note: When **Validate Target Table** is set to **Y**, Data Express performs validation before processing any tables to ensure that the structures of your source and target data stores are exactly the same. If the structures differ and **Always Drop Table** is set to **N**, the method does not execute and an error message displays instructing you to change your **Always Drop Table** setting in order to proceed with the creation of your test environment.

CSV Data Store

Controls whether the tables referenced by a given method should be stored in text files delimited by commas. To enable, change the **CSV Data Store** parameter from **N** to **Y**.

COBOL Masking

Controls whether you are using COBOL routines to mask your data. Possible values are:

Y	By default, **COBOL Masking** is set to **Y**.
N	If you want to use C masking routines, change the **COBOL Masking** parameter to **N**.
O	When the value is "O" (from Others), the product can use any language satisfying the following requirements: • Be built to a callable shared object (UNIX), or DLL (Windows), with the relevant entry points exposed/ exported as required. • Be callable from C. • Have the variables coded precisely as per the COBOL copybook UTEUSFLW.CPY (noting that the content/ structure differs slightly from the mainframe in order to more closely mimic C data types), both regarding data type and size. • Cope with platform-specific byte ordering of numeric data. The C engine code and COBOL copybook expect all numeric parameters to be passed in machine-specific byte ordering. (For example, Intel platforms such as Linux and Windows use the opposite byte ordering to UNIX platforms such as HP-UX, Solaris and AIX).

Validate Target Table

Controls whether the target tables are validated against the source tables to ensure that the structures are exactly the same. By default, **Validate Target Table** is set to **Y**. If you do not want to validate the table structures , change this parameter from **Y** to **N**.

Note: You may not want to validate the target table because if the validation fails (perhaps due to differing column numbers) dropping the table and then recreating the table will create the table in a different tablespace. Set the **Validate Target Table** parameter to **N** only if you are the target and source tables match.

Validate Source Schema

Controls the usage of source schema. If **Y**, the target schema name is the same as the source one. If **N**, the default target schema name is used. By default, **Validate Source Schema** is set to **Y**.

Enable Logging

For more information about logging, see *Logging Exit Routines* in the *Data Masking Guide*.

Hide Progress

Suppresses visual progress during validation.

Hide Log Message

Controls the display of log messages in the `dxe_errorlog.txt` file when **Enable Logging** is used.

Use Extended C Masking

Controls the use of extended C masking. If **N**, standard masking is used. This the default. If set to **Y**, extended C masking is used. For information on the differences between standard C masking and extended C masking, see *Appendix B* of the *Data Masking Guide.*

Trim Spaces for ODBC Char Columns

Optional. If **Y**, final spaces in ODBC CHAR columns are replaced with low-values (x00). If **N** or the **Trim Spaces for ODBC Char Columns** parameter is omitted, final spaces in ODBC CHAR columns are left unchanged.

Commit Size(MBs) [user range = from 1 to 4000, default range = 0]:0

Optional. Controls the frequency of commits, and can be used to process very large numbers of inserts within the same transaction.

0	A commit is performed immediately after processing 95 MB; one commit for each 95 MB processed. Default.
n	A numerical range from 1 to 4000 indicating the number of processed megabytes (MBs) after which to perform each commit. Any number over 4000 is interpreted as 4000.

Parse String finding LOW-VALUES

Determines whether or not a column is masked based on its use of LOW-VALUES.

N	When the first character of a column set to be masked is LOW-VALUES, the column is not masked. Default.
Y	When the column contains any characters other than LOW-VALUES, it is masked.

Create Table using CHAR clause

When the DROP TABLES option is used, determines whether to create a table using the CHAR data type for CHAR columns only, or to create it using the CHAR data type for both CHAR and VARCHAR columns.

N	When the DROP TABLES option is used, the CREATE TABLE statement uses the CHAR data type for CHAR columns only. Default.
Y	When the DROP TABLES option is used, the CREATE TABLE statement uses the CHAR data type for VARCHAR columns.

Execute script when the process aborts

Determines whether or not to execute pre and post scripts when the process aborts.

N	Pre and post scripts are not executed when the process aborts. Default.
Y	Pre and post scripts are executed when the process aborts.

Execute disable/enable CONSTRAINTS

Determines whether or not Data Express provides ENABLING/DISABLING SQL commands to preserve existing CONSTRAINTS relationships.

N	No constraints are created. Default.
Y	DISABLE/ENABLE constraints are created, but only when **Always Drop Table:N** is set.

Execute Aborted Tables

Determines whether or not the engine logs all successfully processed data stores or only the failed data store elaborations from the previous run.

N	The engine works normally, and logs the successfully processed data stores. Default.
Y	The engine processes only the failed data store elaborations from the previous run.

The following settings apply only to the Oracle Extension:

Text Output

By default, Data Express inserts the processed data into the specified target data store. You can configure Data Express to write the output to a text file instead, which can be subsequently loaded into an Oracle database using the Oracle **sqlldr** routine. To enable this, change the **Text Output** parameter from **N** to **Y**.

Data Separator

Oracle Extension only. By default, the delimiter used to separate Oracle fields is the comma (**,**). You can override this default by specifying one or more characters.

Fields Enclosed By

Oracle Extension only. By default, strings are enclosed by double quotation marks (**"**). You can override this default by specifying one or more characters.

If the target table does not exist, the Extension Technology executes a CREATE TABLE statement and then INSERT operations in order to create it. However, if the target table already exists, the way the Extension Technology behaves depends on the settings specified in the `dxeconfig.cfg` file. For example:

Always Drop Table	Validate Target Table	Resulting Behavior
Y	n/a	A DROP TABLE statement is executed before the CREATE TABLE statement and INSERT operations.

N	**Y**	If the target table's metadata matches the source table's metadata exactly, the method will be executed. If any mismatch in the metadata is found, the method will not be executed.
		For example, if the source table is defined as:
		`CREATE TABLE SRC.MYTABLE(C1 INT, C2 CHAR(5), C3 DECIMAL(5,2))`
		and the target table exists, but is defined as:
		`CREATE TABLE TGT.MYTABLE(C3 DECIMAL(5,2), C1 INT, C2 CHAR(5))`

Always Drop Table	Validate Target Table	Resulting Behavior
		the method will not be executed. **Note:** The The `Always Drop Table` parameter set to `N` and the `Validate Target Table` parameter set to `Y` are default settings.

N	**N**	The Extension Technology will not perform any validation. If the target table does not include all columns from the source table (regardless of order), the INSERT operation will fail and the elaboration will terminate abnormally.
		For example, if the source table is defined as:
		` CREATE TABLE SRC.MYTABLE(C1 INT, C2 CHAR(5), C3 DECIMAL(5,2))`
		and the target table exists, but is defined as:
		` CREATE TABLE TGT.MYTABLE(C3 DECIMAL(5,2), C1 INT, C2 CHAR(5))`
		the method should execute and complete normally.
		If the target table is instead defined with a column `C5` instead of `C3`:
		`CREATE TABLE TGT.MYTABLE(C5 DECIMAL(5,2), C1 INT, C2 CHAR(5))`
		the INSERT statement will fail, causing the method to terminate abnormally.

Identify Externally

Oracle Extension only. Identifies connection criteria for connecting to source and target databases.

Y	Data Express uses the information in the configured Wallet rather than the `method.rc` file to connect to the database, eliminating the need to specify user name and password credentials.

N	Database connection requires that you specify a user name and password.

Using append condition

Optional. Oracle Extension only.

Y	During the execution of the **dxestart** module, adds the APPEND condition to the INSERT statement and the NOLOGGING condition to the CREATE TABLE statement. ⚠ **Important:** For the full benefit, you must also set **Always Drop Table** to **Y**.
N, or the **Using append condition** parameter is omitted	Nothing is added during the execution of the **dxestart** module.

Commit Size

Optional.

0	Commit is performed after processing all inserts. Default.
n	A numerical range from 1 to 4000 indicating the number of processed megabytes (MBs) after which to perform a commit. This enables incremental commits and should be used when processing a very large number of inserts.

Data Store SQL Scripts

Each data-store-specific SQL script contains the following types of information:

- Data Express configuration options denoted with the prefix **DX:**
- SQL statements
- Comments denoted with the character prefix --

Filenames

The file names used by Data Express to execute data-store-specific operations are:

- **pre_*ConnectionAlias*.sql** - executes or sets options prior to Data Express processing
- **post_*ConnectionAlias*.sql** - executes or sets options after Data Express processing

  where *ConnectionAlias* specifies:

  - the DSN (data source name) for the ODBC Extension
  - the Oracle TNS service name for the Oracle Extension

For example, with the ODBC Extension, if a given table is accessed via the **MYSOURCE** DSN and the target data store for that table is **MYTARGET**, the configuration file names would be named:

- **pre_MYSOURCE.sql**
- **post_MYSOURCE.sql**
- **pre_MYTARGET.sql**
- **post_MYTARGET.sql**

Configuration Options

Controlling configuration at the data store level allows data-store-specific SQL operations to be performed during before or after Data Express processing.

The following pre-processing configuration settings can be specified:

- **IdentifierDelimiter**

  Specifies two characters to be used as an opening delimiter and a closing delimiter when the data store type requires that the schema name, table name, and column name identifiers needed to be delimited. For example, if working with Microsoft SQL Server data stores, the value would need to be set to `DX:IdentifierDelimiter:[]`
- **IdentityInsertOn**

  Specifies to maintain the identity values from the source database in the target database when the data store type requires that this functionality be enabled. Enabling this parameter causes a SET IDENTITY_INSERT statement to be executed, which will allow values to be inserted into the identity column of a given table. For example, when working with Microsoft SQL Server, `DX:IdentityInsertOn:SET IDENTITY_INSERT ~ ON` results in `SET IDENTITY_INSERT MYSCHEMA.MYTABLE ON`
- **EmptyTableCommand**

  By default, if the **Always Drop Table** parameter is set to **N** within **dxeconfig.cfg** and the table being processed already exists within the specified schema, the table contents will always be emptied prior to inserting processed values. (Data Express accomplishes this by executing the **DELETE FROM** statement, which removes rows from your table one at a time.) You can override this default by choosing a different option that allows your data store to perform more optimally such as the option **TRUNCATE TABLE**, which results in the execution of the TRUNCATE TABLE statement being executed. For example, `DX:EmptyTableCommand:TRUNCATE TABLE` results in `TRUNCATE TABLE` *`MYSCHEMA`*`.`*`MYTABLE`*

The following post-processing configuration setting can be specified:

- **IdentityInsertOff**

  Specifies the default behavior of using the automatically generated value for the identity column in the target database, which may cause result in differing identity values for the source and target databases. Disabling this parameter causes a SET IDENTITY_INSERT statement to be executed. For example, when working with Microsoft SQL Server, `DX:IdentityInsertOn:SET IDENTITY_INSERT ~ OFF` results in `SET IDENTITY_INSERT` *`MYSCHEMA`*`.`*`MYTABLE`* `OFF`

Sample Scripts

For your convenience, some sample scripts containing some of the options described in *Configuration Options* are provided in the **config** directory as templates in order to assist you in building your specific configuration:

- **pre_Oracle.sql**
- **post_Oracle.sql**
- **pre_Microsoft SQL Server.sql**
- **post_Microsoft SQL Server.sql**

DXCOMMON Environment Variable

Applies only when using the Oracle Extension on a Windows platform.

During the extension elaboration process, the Oracle extension creates a temporary text file, `dxe_common.txt`, to track the execution chain. It also deletes this file at the end of the extension elaboration process. Because `dxe_common.txt` is created in the `log` subdirectory of the Data directory, when that directory is on a network share, the process of creating and deleting this file can degrade overall performance. Therefore, if you have changed the default Data directory for extension technology to a shared directory on a network, set the DXCOMMON environment variable to improve performance.

The DXCOMMON environment variable enables you to specify a local directory in which to create the `dxe_common.txt` file.

To implement this environment variable, on a machine that runs Data Express, create a new system variable, DXCOMMON, and set its value to a local directory. For example:

```
DXCOMMON=C:\dxecommon
```

If the specified directory does not exist, the Oracle extension creates it when it creates the `dxe_common.txt` file for the first time.

Extension Technology Directories

For the Extension Technology, data files and executable files are stored in separate locations. This section describes these directories.

Executables

The following table shows the default directory containing your Extension Technology executable:

OS	ODBC Extension	Oracle Extension
Windows	`C:\Program Files\Micro Focus\Data Express 4.0\odbc`	`C:\Program Files\Micro Focus\Data Express 4.0\oracle`
UNIX	`/opt/microfocus/dataexpress/odbc`	`/opt/microfocus/dataexpress/oracle`

Note: The default data and executable directory for UNIX happens to be the same.

You can install to a different root directory if desired.

Data

The data directory is used for data store configuration files and log files.

The following table shows the default data directory needed for the ODBC Extension or Oracle Extension depending upon your platform:

OS	ODBC Extension	Oracle Extension
Windows	`%ALLUSERSPROFILE%\Micro Focus\Data Express 4.0\odbc`	`%ALLUSERSPROFILE %\Micro Focus\Data Express 4.0\oracle`
UNIX	`/opt/microfocus/dataexpress/odbc`	`/opt/microfocus/dataexpress/oracle`

The path specified for the `%ALLUSERSPROFILE%` environment variable differs depending on your Windows platform. For example:

- Windows XP - `C:\Documents and Settings\All Users\Application Data\Micro Focus\Data Express 4.0`
- Windows Vista - `C:\ProgramData\Micro Focus\Data Express 4.0`

Note: When using the Oracle Extension on a Windows platform, if you choose to change the default Data directory to a shared directory on a network, this could affect performance. For information on improving performance in this scenario, see *DXCOMMON Environment Variable*.

config

The files generated by the Distributed Exporter and the Distributed Sampler must also be stored in the `config` subdirectory of the data directory as this is necessary for Extension Technology execution.

If you are invoking the ODBC Extension or the Oracle Extension from a Windows environment, you can specify the `config` subdirectory as the output directory for masked, reduced, or sampled data stores. Otherwise, you can move the files into the default directory manually.

log

After the execution of the ODBC Extension or the Oracle Extension, a `dxe_errorlog.txt` log file is created within the `log` directory in a subdirectory named: *`YYYY-MM-DD_hh-mm-ss`*. For example:

```
log\2009-03-16_19-10-08\dxe_errorlog.txt
```

If errors occur during processing, the error log displays automatically.

If the execution of the ODBC Extension or the Oracle Extension includes subset extraction processing, `dxe_summarylog.txt` and `dxe_statistics.txt`. The Oracle Extension also creates `dxe_common.txt` to track the execution chain; however, this file is deleted at the end of the extension elaboration process. Log files are also created in the same subdirectory.

If the execution includes sampling analysis, the `sampling.log` file is also created in this subdirectory; however, this file is used to load sampling results into Data Express and is not meant to be human-readable.

Invoking Extension Technology

The *Data Express for Distributed Systems Tutorials* demonstrates the default method for executing Extension Technology on a Windows platform using default mode.

This section walks you through configuration settings necessary for the execution of Extension Technology.

To invoke the Extension Technology:

1. Copy the files generated by the Distributed Exporter from the output directory you specified, and paste them in the **config** subdirectory of the appropriate default data directory on Windows.

   Note: For a list of default directories, refer back to the section *Data* of the chapter *Extension Technology Directories*.
2. Ensure that your data store environment is configured appropriately. If you are working with ODBC data stores, an ODBC Data Source Names (DSN) should be configured as required for your source and target data stores. If you will be executing the ODBC Extension from a UNIX environment, the ODBC driver manager and driver environment should be configured. Typically, this includes:

   - The **ODBCINI** environment variable should point to the ODBC configuration file.
   - The setting of the **LIBPATH** (AIX), **SHLIB_PATH** (HP/UX), or **LD_LIBRARY_PATH** (all other flavors of UNIX) environment variable should include the **lib** folder under your ODBC Driver Manager installation.
   - Implement other settings as defined by your ODBC/ODBC Driver Manager documentation.

     Note: If you will be executing the ODBC Extension from a UNIX environment, make sure that the UNIX ODBC DSN matches the ODBC DSN specified when you did the export on Windows. If you are making use of C or COBOL masking routines, verify that you have a development environment for building masking routines and a run-time environment for executing masking routines.
3. Verify the Run-time Knowledge Base has been created and is configured with the appropriate privileges. For more information, see the sections *Run-time Knowledge Base Requirements* and *User Credential Requirements* of your ***Readme***.
4. Navigate to the executables directory, which contains the Extension Technology (ODBC Extension or Oracle Extension).
5. Execute the appropriate command to invoke the Extension Technology:

Mode	Windows	UNIX
Default	**dxestart**	**./dxestart**
Parallel	**dxestart -d *DirectoryName***	**./dxestart -d *DirectoryName***
Unload	**dxeunload**	**./dxeunload**

Tip: `dxeunload` has the same syntax and format as `dxestart`. See *The dxestart Utility* for more information on using `dxestart`.

The Extension Technology processes all the tables previously specified within Data Express, and reports a status for each data store, for example:

```
 Data Express Distributed Server 4.0.0
c) Micro Focus (IP) Limited 2003-2009
Table xdb.SRC.CUSTOMER: 21 rows [##########] Reduced to 6 rows.
Table xdb.SRC.ORDERS: 5 rows [##########] Reduced to 3 rows.
Table xdb.SRC.ITEMS: 8 rows [##########] Reduced to 5 rows.
Data Express processing completed OK.
```

6. Use the Distributed Statistics Loader to populate run-time statistics into Data Subset Extraction. For more information, see section *Distributed Statistics Loader* of chapter *Extension Technology Utilities*.

The dxestart command

`dxestart` includes parameters for passing connection information about the input and output data stores. This is useful if you have strong security requirements and use scripts for batch processing.

Parameters for `dxestart` are valid for both Oracle and ODBC extensions.

Syntax

```
dxestart -d <path> -i<input method.rc> -o<output method.rc>
         [-f <from-step> -t <to-step>]
```

Parameters

`-d <path>` specifies that Data Express reads the `method.rc` location specified by `<path>`. If not specified, Data Express uses the default configuration folder. For example:

```
dxestart -d "C:\Testlib1"
```

This example reads the connection information from the `method.rc` file in `C:\Testlib1\config` and creates the log and statistic files in `C:\Testlib1\log`.

`-i <input method.rc>` reads the input connection information from a different `.rc` file (named `[FileName].rc`, where `[FileName]` is a name you specify for the file). This file is the functional equivalent of the `method.rc` file. Specify the name and the path to the new `.rc` file in this parameter.

`-o <output method.rc>` reads the output connection information from a different `.rc` file (named `[FileName].rc`, where `[FileName]` is a name you specify for the file). This file is the functional equivalent of the `method.rc` file. Specify the name and the path to the new `.rc` file in this parameter. For example:

```
dxestart -d "C:\Testlib1" -i "C:\Testlib2\method1.rc" -o "C:
\Testlib3\method2.rc"
```

This example reads the input connection information from the file `method1.rc` that resides in `C:\Testlib2`, the output connection information from the file `method2.rc` that resides in `C:\Testlib3`, and the exported configuration information from `C:\Testlib1\config`. It then creates the log and statistic files in `C:\Testlib1\log`.

See *Creating an Encoded Connection File* for instructions on creating the input and output connection file that can be used with the `-i` and `-o` parameters.

`-f <from-step> -t <to-step>` enables you to specify a range of steps to be executed. `<from-step>` represents the first step of the range, and `<to-step>` represents the last step of the range.

Execution Modes

This section describes in greater detail the three Extension Technology Execution modes: default, parallel execution, and unload.

Default Mode

Use the default execution mode when the source data store meets all the schema requirements described in the section *Schema Requirements* of chapter *Requirements* of the ***Readme*** .

Parallel Mode

Use the parallel mode when you want to execute more than one method simultaneously.

Warning: Use of the parallel mode where multiple methods access the same tables within the target data store(s) is not currently supported.

In order to run a parallel execution, the files created after a successful export from Data Masking or Data Subset Extraction must be stored in a different data directory for each method.

1. Create a new directory for a method.
2. Within the newly created directory, create the subdirectory **config**.
3. Copy the contents of the default **config** directory into the newly created **config** directory.

   For more information about the default **config** directory, see the section *Extension Technology Directories* in this guide.
4. Export or manually move the files generated by the Distributed Exporter into the new **config** directory.

   Note: Bear in mind that the machine where the ODBC Extension is executed may not be the same the same machine where the Windows client software is running.
5. Use the **dxestart** command with the **-d** option to specify the new location.

   See *Invoking Extension Technology* for more information.

Note: You can execute the first method in default mode, which utilizes your default data directory, but you must execute all subsequent methods in parallel execution mode.

For more information about default directories, see the section *Data* in the chapter *Extension Technology Directories*.

Unload Mode

In unload mode, the same schema requirements apply (as described in the section *Schema Requirements* of chapter *Requirements* of ***Readme***) , but with a difference: Data Express does not use privileges against the source data store. Instead, information is unloaded from the source data store to separate *staging* data store where these privileges are can be authorized.

You can use unload mode for the following two scenarios:

Source Data Store	Staging Data Store	Target Data Store
Data Store A	Data Store B	Data Store C
Data Store A	Data Store C	Data Store C

These examples show that the staging data store may or may not be the actual target data store.

For example, to use unload mode with Data Masking:

1. In the Distributed Loader within Data Builder, specify your source data store as the **Source database** (as normal), and upload tables.

   For more information about the Distributed Loader, see the section *Using Distributed Loader* of *Part 2. Using Distributed Data Stores with Data Express.*
2. In Data Masking, enable masking of all the tables catalogued.

   For more information, see *Enable Masking* within the section *Data Masking* of *Part 4: Data Express for Distributed Data Stores Tutorials*.
3. In the Distributed Exporter within Data Masking, specify your staging data store as your **Target database**, and then export files.
4. Copy the exported files into the **config** directory.

   Note: For more information about default directories, see the section *Data* in the chapter *Extension Technology Directories*.
5. Execute the Extension Technology using unload mode.

   For more information, see the section *Invoking Extension Technology*.

   The data from the source data store is extracted and inserted into the staging data store after the required tables were created.
6. From this point forward, you can process your data normally provided that you specify the staging data store as your source and the target data store as your target. This means that you load from staging and export into target.

Note: A similar approach can be used to unload with Data Subset Extraction; after using the Distributed Loader to upload your tables, you can mark all the tables as "register tables". Then, it is very simple to create a subsetting method of your register tables from Data Subset Extraction. Use the Distributed Exporter to copy the exported files into the **config** directory, and execute the Extension Technology using unload mode.

Error Diagnosis

Diagnostic information including error messages generated during the execution of the ODBC Extension or Oracle Extension is written to log files.

These messages are used by administrators and Micro Focus Product Support representatives.

Extension Technology Log Files

The Data Express Extension Technology generates three log files: `dxe_errorlog.txt`, `dxe_summarylog.txt`, and `dxe_statistics.txt`, which are stored in a subdirectory of the `log` directory. For details, see *Extension Technology Directories*.

dxe_errorlog.txt

The `dxe_errorlog.txt` file contains some processing information as well as information on any errors that have occurred during the processing. The level of detail displayed in this log is controlled by the LogLevel configuration option.

A sample of the `dxe_errorlog.txt` file with LogLevel set to 1 is shown:

```
 Initialization: populating Data Express Knowledge Base

-----------------------------------------------------------------------------

Processing table GSDEMO.SRC.CSTOMER
SQLCODE=-28,SQLSTATE=S0002:[XDB][Micro Focus XDB ODBC 32-Bit Driver]
```

```
[GSDEMO]X28
Cannot find table/view/synonym 'SRC.CSTOMER'.
DE-0074: Unable to SELECT from table in the source database.
Ensure that the database permissions provide the 'install' user with SELECT
 privileges.
 Summary: TableID=2 TableName=GSDEMO.SRC.CSTOMER TotalCount=0 ReducedTo=0
CpuTime=
170340.391000 ElapTime=0.000000

-----------------------------------------------------------------------------

Processing table GSDEMO.SRC.EMPLOYEE
Summary: TableID=4 TableName=GSDEMO.SRC.EMPLOYEE TotalCount=10 ReducedTo=10
CpuTime=1.153000 ElapTime=0.000000
```

A sample of the `dxe_errorlog.txt` file with LogLevel set to 3 is shown:

```
 Initialization: populating Data Express Knowledge Base
 -------------------------------------------------------------------
Processing table GSDEMO.SRC.CSTOMER
dxestart: executing dxeinit GSDEMO.SRC.CSTOMER | dxeextracter GSDEMO.
CSTOMER 1 1 28216 "DATAEXPRSS" 1 1 1 | UDCZIP5 GSDEMO.SRC.CSTOMER |
dxewriter GSDEMO.SRC.CSTOMER 'N' 1 | dxefinal 0 1 2 "DATAEXPRSS"
"DataExpress_Business_Protection" '0' 'N' 1
SQLCODE=-28,SQLSTATE=S0002:[XDB][Micro Focus XDB ODBC 32-Bit Driver]
[GSDEMO]X28 Cannot find table/view/synonym 'SRC.CSTOMER'.
DE-0074: Unable to SELECT from table in the source database.
Ensure that the database permissions provide the 'install' user with
SELECT privileges.
Summary: TableID=2 TableName=GSDEMO.SRC.CSTOMER TotalCount=0
ReducedTo=0 CpuTime=170340.406000 ElapTime=0.000000
 -------------------------------------------------------------------
Processing table GSDEMO.SRC.EMPLOYEE
dxestart: executing dxeinit GSDEMO.SRC.EMPLOYEE | dxeextracter
GSDEMO.SRC.EMPLOYEE 1 1 28216 "DATAEXPRSS" 1 1 1 | UDCZIP5
GSDEMO.SRC.EMPLOYEE | dxewriter GSDEMO.SRC.EMPLOYEE 'N' 1 |
dxefinal 1 1 4 "DATAEXPRSS" "DataExpress_Business_Protection"
 '0' 'N' 1
Extracter: executing SELECT DSE_TABLE.* FROM SRC.EMPLOYEE DSE_TABLE
Writer: executing DROP TABLE TGT.EMPLOYEE
Writer: executing CREATE TABLE TGT.EMPLOYEE (E_NO SMALLINT
NOT NULL, LNAME CHAR(10) NOT NULL, FNAME CHAR(10) NOT NULL, STREET
CHAR(20), CITY CHAR(15), ST CHAR(2), ZIP CHAR(5), DEPT CHAR(4),
PAYRATE DECIMAL(15, 2), COM DECIMAL(2, 2))
Writer: executing INSERT INTO TGT.EMPLOYEE (E_NO, LNAME, FNAME,
STREET, CITY, ST, ZIP, DEPT, PAYRATE, COM) VALUES (?, ?, ?, ?, ?, ?,
?, ?, ?, ?)
Summary: TableID=4 TableName=GSDEMO.SRC.EMPLOYEE TotalCount=10
ReducedTo=10 CpuTime=1.138000 ElapTime=0.000000
```

Note: Following an abnormal termination, Data Express prompts you to view the `dxe_errorlog.txt` file. Answer **Yes** at the prompt to view the file in Notepad (Windows) or cat file | more (UNIX).

dxe_summarylog.txt

The `dxe_summarylog.txt` file provides a brief summary of the data stores processed during subset extraction with statistics on the number processed and processing time.

A sample of the `dxe_summarylog.txt` file is shown:

```
 Machine ID:3 Method:SIMPLE TestRun:'N' Date:20090302
Description:SIMPLE
 STEP   TABLEID   TABLE NAME          ROWS      CPU TIME  ELAPSED TIME
                                ORIGINAL   OUTPUT
 --------------------------------------------------------------------------
 10      2         xdb.SRC.CUSTOMER    21   6    1.23      1.00
 20      6         xdb.SRC.ORDERS       5   3    0.80      0.00
 30      5         xdb.SRC.ITEMS        8   5    0.75      1.00
```

dxe_statistics.txt

The `dxe_statistics.txt` file provides a brief summary of the data stores processed during subset extraction similar to the `dxe_summarylog.txt`. However, the `dxe_statistics.txt` file is used by the Distributed Statistics Loader to populate run-time statistics into Data Subset Extraction and is not intended for viewing even though it is human-readable.

Masking Routines

A masking routine is a program that contains the processing and masking logic needed to perform the actual masking on your data.

The masking routines for the Extension Technology can be written in C or COBOL or any other type that uses the same interface and structure as the COBOL exits.

Note: This section is only intended to provide additional information regarding Extension Technology masking routines for the Micro Focus Data Express for Distributed Systems solution. Use the ***Data Masking Guide*** as your definitive Data Masking resource.

The ODBC Extension ships with pre-built masking routines, as documented in the ***Data Masking Guide*** .

Using Predefined Masking Routines

A number of masking routines are provided with your Data Express installation. For your convenience, these routines are built and ready to use. The source files are provided in the **c** and **cobol** subdirectories of **src** within your Data Express installation directory, along with template sources to assist in the creation of new routines.

C Masking Routines that Access Replacement Lookup Data

The following C routines require access to a data store that contains replacement lookup data:

- UDCADDC
- UDCCDFC
- UDCCFPI
- UDCCOMC
- UDCNAMC
- UDCSNC2
- UDCSURC

Note: To use C masking routines that access replacement lookup data:

1. Create and populate the lookup tables.

   Data Express includes the SQL script **kb.sql** that can be used to create and populate the lookup tables with predefined lookup data that works with the routines listed previously. The script may need to be amended to suit the requirements of your data store. For example:

   - The size of the replacement lookup data needs to be sized appropriately so that the masked value will fit into your user table. For example, if the column being masked is defined as a

CHAR (30), you must ensure that the replacement lookup data is no larger than 30 characters in length.

- Your data store may not require semi-colon terminators for SQL statements
- If the data store housing your lookup data is an Oracle database, you will need to change the data type **VARCHAR** to **VARCHAR2**.

2. Configure Data Express so that it can access the data store housing your lookup data by editing **hdblayer.ini**, which is located within the executables directory of your installation.

   Note: No **hdblayer.ini** configuration is needed if you create lookup tables in the same place where you create the runtime knowledge base. Edit information in the file as follows:

   ```
   RDBMS = ConnectionType
    DBNAME = ConnectionAlias
       DBOWNER = TableOwner
       USER = UserID
    PSWD = Password
   ```

   where:

   - *ConnectionType*: Set to **ODBC** for the ODBC Extension or **ORACLE** for the Oracle Extension.
   - *ConnectionAlias*: Set to the data store containing the lookup data.

     For the ODBC Extension, this is the data source name (DSN).

     For the Oracle Extension, this is the Oracle TNS service name as specified in your **tnsnames.ora** configuration file.
   - *TableOwner* - The table qualifier for the lookup tables.
   - *UserID & Password* **-** The connection criteria for the data store containing the lookup tables.

     For example:

     ```
      RDBMS = ODBC
     DBNAME = MYDSN
     DBOWNER = MYOWNER
     USER = MYUSER
     PSWD = MYPASSWORD
     ```

3. Encrypt the **hdblayer.ini** file by executing the appropriate command.

OS	Command
Windows	**dxeencrypt hdblayer**
UNIX	./ **dxeencrypt hdblayer**

C

The source files are provided in the **c** and **cobol** subdirectories of **src** within your Data Express installation directory, along with template sources to assist in the creation of new routines.

Creating C Masking Routines

To create a C masking routine:

1. Make a copy of the template masking routine **UDCDEMO.c** file.
2. Update the **CHANGER_NAME** definition by replacing the name **CHANGER** with the name of your masking routine:

```
 #define CHANGER_NAME "CHANGER" /* Apply changer's name here! */
```

3. Review the processing logic in the **ProcessColumns()** function in order to verify that the data type is correct and that the masking routine specified for that data element is in this particular routine:

```
 /* Data processing example */
if ((pData->nType[i] == TYPE_STRING ||
pData->nType[i] == TYPE_INT_STR ||
pData->nType[i] == TYPE_CLOB) &&
strcmp(pTable[i].szChangerProgram, CHANGER_NAME) == 0)
{
pData->uData[i].szData = ProcessString(pData->uData[i].szData, &pTable[i])
;
}
```

4. Code your masking logic within the **ProcessString()** function.
5. Edit the **makefile**:

   - Add an entry to the **CHANGERS** macro for your new masking routine.
   - Apply the appropriate rules for building your masking routine:

   For your Windows **makefile**, copy the entry below and then replace **UDCDEMO** with the name of your masking routine:

```
 UDCDEMO.obj: dxe_common.h $(COMMON_HEADERS) UDCDEMO.c
    !$(CPP) $(CFLAGS) $*.c
```

   For your UNIX **makefile**, copy the entry below and then replace **UDCDEMO** with the name of your masking routine:

```
 UDCDEMO: libdecommon.a UDCDEMO.o
    $(CC) $@.o $(LDFLAGS) -o $@
```

Building C Masking Routines

On Windows platforms, build your C masking routine by invoking **nmake** from a Visual Studio command prompt. On UNIX platforms, build your C masking routine by invoking **make**.

COBOL

In order to work with COBOL masking routines, you need to have the COBOL environment configured:

- For building the routines, you need the COBOL development environment.
- For executing the routines, you need the COBOL run-time environment only.

Building COBOL Masking Routines

To create a COBOL masking routine:

1. Make a copy of the template masking routine **UDCDEMO.CBL**.
2. Update the **PROGRAM-ID** clause to include the name of your masking routine.
3. Insert your masking logic within the PROCEDURE DIVISION, after the following comment:

```
 *
  * Insert processing logic here.
     *
```

4. Update the build script to add instructions for building the new module. For Windows platforms:

Step	Action	Details

1	Update the **build.bat** script.	Add the appropriate COBOL command for generating the source to object code (.**obj**), for example:
		`cobol`
		`MYMODULE`
		`.CBL DEFAULTBYTE"00"`
		`CASE:`
		Note that routines should be compiled with the **DEFAULTBYTE "00" CASE** compiler directives.
		Update the setting of the **CHANGERS** variable to add your masking routine name.

2	Update **DE_CHANGERS.DEF**.	Add an entry for your masking routine at the end, using a unique value, for example: *`MYMODULE`* `@101`

For UNIX platforms, update the **build** script by adding the appropriate COBOL command for generating the source to object code (**.o**), for example: `cob -xc` *`MYMODULE`*`.CBL -C "DEFAULTBYTE(00) CASE"`

Building COBOL Masking Routines

On Windows platforms build your COBOL masking routine by invoking **build**. On UNIX platforms, build your COBOL masking routine by invoking **./build**.

Extension Technology Utilities

This section describes utilities provided with the ODBC Extension and Oracle Extension.

To use the utilities, you need the connection alias, username, and password for your data store.

dxegenloadfile

If your Windows machine does not have direct access to your data store, data can be loaded into the Data Express Knowledge Base from files. The `dxegenloadfile` utility works with the Distributed Loader to load table metadata and referential integrity information from files into the Knowledge Base.

The syntax is:

```
dxegenloadfile -c - d<path> -i<input method.rc>
```

where:

`-c` specifies that there is no console interaction. The information about the connection comes from the standard `method.rc` file. It contains the connection information including database name.

If `-c` is the only parameter specified, Data Express reads the `method.rc` file from the default configuration folders. The `dxegenloadfile` utility generates files which are written to the default configuration folders.

`-d <path>` specifies Data Express reads the `method.rc` file from the folder indicated by `<path>`. For example:

```
dxegenloadfile -d "C:\Testlib1\config"
```

This example reads the connection information from the `method.rc` file in `C:\Testlib1\config` and creates the list and log file in `C:\Testlib1\log`.

If not specified, Data Express uses default configuration directories.

Note: The `-d` and `-c` parameters are mutually exclusive. If `-d` is used, then `-c` must not be specified.

`-i <input method.rc>` reads the input connection information from a different `.rc` file (named `[FileName].rc`, where `[FileName]` is a name you specify for the file). This file is the functional equivalent of the `method.rc` file. Specify the name and the path to the new `.rc` file in this parameter. For example:

```
dxegenloadfile -d "C:\Testlib1" -i "C:\Testlib2\method1.rc"
```

This example reads the connection information from the file `method1.rc` that resides in the `C:\Testlib2` folder creates the file list and log file in the folder `C:\Testlib1\log`.

See *Creating an Encoded Connection File* for instructions on creating the input connection file that can be used with the `-i` parameter.

To load data from files, you must first install the ODBC Extension or Oracle Extension and then run the `dxegenloadfile` utility to generate the text files **DataStoreLoader.txt** and **RILoader.txt**, which are placed into the log folder.

For more information about the Distributed Loader, see section *Load from File* in *Part 2. Using Distributed Data Stores* in this guide.

dxelc

The `dxelc` utility is designed to update the knowledge base, changing data structures from a previous release. The input of this command is the output of `dxegenloadfile`.

The syntax is:

```
dxelc -d "<lcmap.rc>"
```

It uses a configuration file that includes the following information in the order shown:

1. Full path to the output generated by the `dxegenloadfile` command
2. Machine ID
3. Company ID
4. ODBC instance name for reference environment
5. Schema name for reference environment
6. ODBC instance name for Data Express
7. Schema name for Data Express

Example:

```
<full_path_DataStoreLoader.txt>,<machine_id.>,
  <company_id.>,<reference_ODBC>,<reference_schema>,
  <kb_ODBC>,<kb_schema>
```

Note: To get `lcmap.rc`, encrypt `lcmap.ini` using `dxeencrypt lcmap.ini`.

dxegentypeinfo

dxegentypeinfo is an ODBC Extension utility used by Micro Focus for diagnostic purposes. The utility generates a file **dxe_odbcdiag.txt** that lists details of ODBC data types supported by the data store, which is placed into the **log** folder.

Distributed Statistics Loader

For the Data Express for Distributed Systems solution, run-time statistics recorded during subset extraction execution are not automatically loaded into your Data Express Knowledge Base (KB).

The Distributed Statistics Loader is a utility that lets you load run-time statistics into the KB. Once you have executed the Extension Technology to subset your data, a log filed called **dxe_statistics.txt** is created. This file is used to load the run-time data into the KB so that the statistics are available in Data Subset Extraction.

Use the Distributed Statistics Loader after you have executed the Extension Technology to subset your data.

To use the Distributed Statistics Loader:

1. Use FTP, NFS, or some means to copy the **dxe_statistics.txt** file generated by the Extension Technology to the client machine where you have the Data Express Window client-side software installed.
2. Start Data Subset Extraction.

   See *Starting Data Subset Extraction* in your ***Data Subset Extraction Guide*** for specific instructions.
3. Click (**Distributed Statistics Loader**).
4. In the **Log file to import** box, specify the **dxe_statistcs.txt** log file name including the path where you saved it on your Windows machine.
5. Click **Check Log File** to verify that the log file you select is correct.

   The remaining fields in the **Distributed Statistic Loader** windows populate from information specified from the **dxe_statistics.txt** file.
6. Click **Load** to load the subset extraction statistics into Data Subset Extraction.

   A *Statistic loaded successfully!* message is displayed.

   You can view the run-time statistics in various places in Data Subset Extraction. For example:

- Viewing the List of Elaborations in Step by selecting the appropriate method from the Main Window
- Viewing the Data Stores Elaboration from the **Elaborations** tab in the Work with Method window
- Viewing the List of Elaborations in Step from the **Method Overview** tab in the Work with Method window

Run-time statistics include the following:

- **Number of Read Records**
- **Number of Written Records**
- **Reduction in Percentage**
- **Simulation Date**
- **Real Extraction Date**
- **Elapsed Time [hh:mm:ss]**
- **CPU Time [hh:mm:ss]**

Note: The **Reduction in Bytes** run-time statistic is not used in the Distributed Systems solution because of the way ODBC-enabled data stores display the record length.

Creating an Encoded Connection File

You can create an encoded connection file by following these instructions:

1. Create a *`Filename`*`.ini` text file that contains at least one line using one of these format specifications:

   `dxegenloadfile (-i option)` or `dxeexport`

   Format specification is `TYPE,ALIAS,SCHEMA,USERID,PASSWORD` where:

   - **TYPE** is ORACLE or ODBC
   - `ALIAS` is either your ODBC DSN or Oracle TNS alias
   - `SCHEMA` is your schema name that contains your tables
   - `USERID` and `PASSWORD` are the user credentials to connect to `ALIAS`

   `dxegenloadfile` example - ODBC,TUTORIAL,SRC,INSTALL,1

   This example uses ODBC extension to connect to DSN TUTORIAL with supplied user credentials, and retrieves table information from the schema SRC, so `dxegenloadfile` can create files for consumption by the Distributed Loader.

   `dxeexport` example - ODBC,Dist KB for DX 4.0,DX40,install,1

   This example uses ODBC extension to connect to DSN Dist KB for DX 4.0 with supplied user credentials, and retrieve knowledgebase information, allowing `dxeexport` to create required files for `dxestart` to mask and subset your data.

   `dxestart (-i option` or `-o option)`

   Format specification is `TYPE,ALIAS,SCHEMA,USERID,PASSWORD` where:

   - `TYPE` is ORACLE or ODBC
   - `ALIAS` is either your ODBC DSN or Oracle TNS alias
   - `SCHEMA` is your schema name that contains your tables
   - `USERID` and `PASSWORD` are the user credentials to connect to ALIAS

   `dxestart -i option` example - ODBC,TUTORIAL,SRC,INSTALL,1

   This example uses ODBC extension to connect to DSN TUTORIAL with supplied user credentials, and is the source data store (input). The source data store is be masked, subsetted, and placed in the target data store.

   `dxestart -o option` example - ODBC,Dist KB for DX 4.0,DX40,install, 1

   This example uses ODBC extension to connect to DSN Dist KB for DX 4.0 with supplied user credentials, and is the target data store (output). The source data store is masked, subsetted, and placed in the target data store.

   Note: For any of these formats, you can optionally add `,,,,,` to the end of the specification.

2. Encode a *Filename*.ini file with dxeencrypt to create an encoded *Filename*.rc file. For example, from a command prompt, enter:

   ```
   dxeencrypt Filename.ini
   ```
3. Use an encoded .rc file with dxegenloadfile, dxeexport, or dxestart.

Batch Distributed Exporter

In addition to invoking the Distributed Exporter directly from the Data Masking or Data Subset Extraction modules themselves, Data Express also provides a batch, command-line utility, dxeexport.

dxeexport

The dxeexport command exports the information required so that a successful masking/subsetting execution can occur when the dxestart command is executed.

Syntax

```
dxeexport [path/]exportparms [path/]connfile
```

Parameters Parameters for dxeeport are valid for both Oracle and ODBC extensions.

path Full path to the specified file; required when the file is not in the current directory.

exportparms The name of the file that contains export parameters. This must be a text (.txt) file. It identifies the group and methods to export from the knowledge base. The text in this file consists of the following items, each delimited by a comma:

- Machine ID
- Company
- Group name
- Method name
- Step range
- Test Run(Y,N)
- Configuration file path (location of your configuration directory)

Note: *ALL is allowed at both the group and at the method level. When *ALL is used, subfolders for each method will be created in the destination directory.

connfile The name of the file that contains encoded connections. This must be an .rc file. dxeexport uses this file to retrieve required information from the Data Express knowledge base. See *Creating a Encoded Connection File* for details.

Example From command line, enter:

```
dxeexport "C:\AA2\Export_Folder\exportparms" "C:\AA2\Export_Folder
\connfile"
```

This command executes a batch export for the method described in exportparms.txt, using the Data Express knowledge base described in connfile.rc.

Assume that exportparms.txt contains one line as follows:

```
PRUEBA,PRUEBA,PG1,PM1,0010, 0100,N,C:\AA2\Export_Folder
```

With this example, the method PM1 of group PG1 on machine PRUEBA and company PRUEBA is exported, from step 10 to step 100, with test run set to No (so real extraction), and the resulting configuration files generated by the process are placed in folder `config` under `C:\AA2\Export_Folder`.

You can now run `dxestart` to mask and subset your data, specifying `C:\AA2\Export_folder` as the `-d` argument.

Data Express for Distributed Data Stores Tutorials

These tutorials introduce the Data Express Windows client and show you how to configure your working environment and start and use the three Data Express modules: Data Builder, Data Masking, and Data Subset Extraction.

Follow the tutorials in this guide in order after you have completed the product installation.

Using Data Express

The easiest way to learn about Data Express is to complete a few simple tasks. This guide takes you through configuring Data Builder, loading metadata into the Data Express Knowledge Base, using Data Masking to implement masking rules on a small amount of data, using Data Subset Extraction to create a simple subset of data, and finally using Data Subset Extraction twice more to create subsets of data using referential integrity. Please note that some of the subsetting examples also mask data, showing that Data Express can do both functions in one method invocation.

The three Data Express modules (Data Builder, Data Masking, and Data Subset Extraction), and the ODBC Extension contain all the tools you need to perform these tasks.

Machine IDs and Company Codes

A Machine ID and Company code pair is essentially an identifier for your Data Express working environment or *workspace*. The Machine ID is the higher-level identifier, while the Company code must be identified by a Machine ID. You can create a new Machine ID and a new Company code for each test environment, or you can create new Company codes under a single Machine ID as needed for each test environment you want to create.

Note: A single machine ID can correspond to numerous company codes. A new company code can be created only if an existing machine ID is first selected.

Sample Data

Data Express includes sample XDB tables for the Data Express Knowledge Base and a corresponding XDB source database called TUTORIAL , which is used throughout the exercises in this tutorial. The database and its sample tables are installed automatically during product installation.

Preparation

In this session, you configure connectivity for the sample XDB data store provided in the product installation and perform setup for basic Data Express options. This involves setting up a data source name (DSN) to enable communication to the XDB TUTORIAL data store for all three client-side modules.

Note: Sessions in this tutorial should be done in order.

Sample Session

In this session, you:

- Configure an ODBC DSN
- Start Data Builder

- Set options

Configuring an ODBC DSN

Data Express requires a user or system DSN for connectivity to your data store through an ODBC driver, which is configured using the **Data Sources (ODBC)** administrative tool.

This section provides instructions for configuring a user DSN needed for the ODBC driver provided with your Data Express installation.

Note: The Data Express product installation will install the correct networking components and ODBC driver.

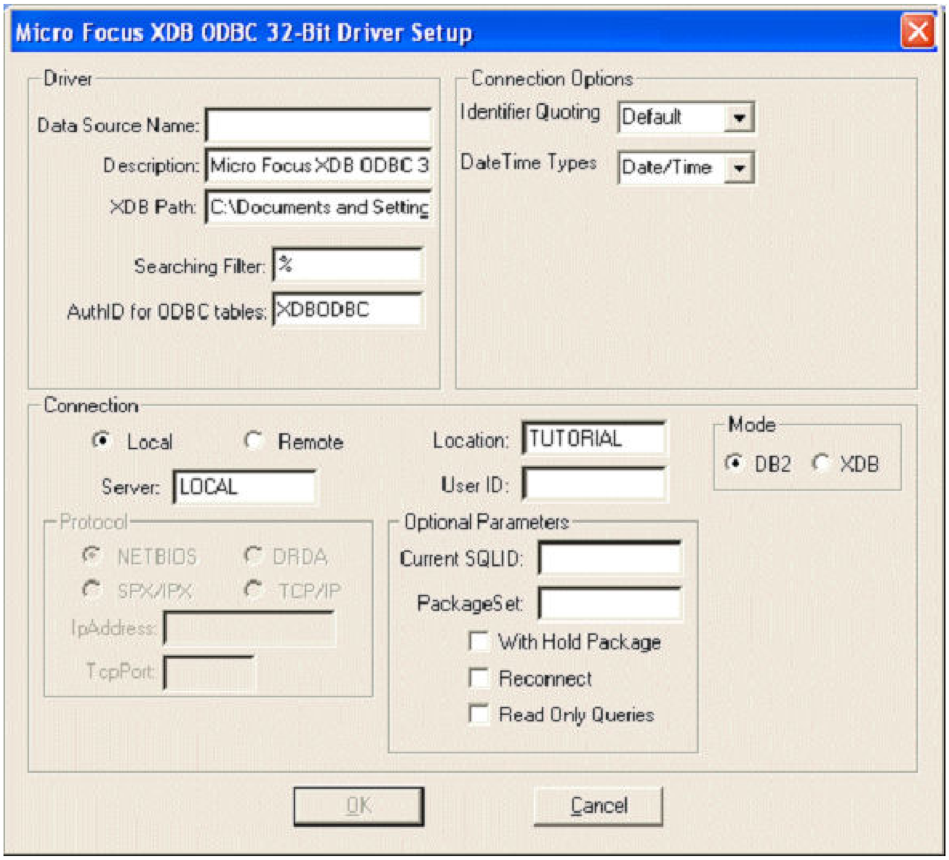
The Data Express Knowledge Base also uses a system DSN for connectivity through an ODBC driver; the driver is installed and the preconfigured DSN is implemented during product installation.

A BDE alias is not required for your ODBC-enabled data store.

To configure your ODBC DSN:

1. Start the ODBC Data Source Administrator. (Use **Start** > **Settings** > **Control Panel** > **Administrative Tools** > **Data Sources (ODBC)**.)

   Note: On some older versions of Windows, you can access Administrative Tools from the **Start** menu.
2. Click the **User DSN** tab.
3. Click **Add**.
4. In the **Create New Data Source** window, select the driver **Micro Focus XDB for DX 4.0**.



Micro Focus XDB ODBC 32-Bit Driver Setup

Driver

Data Source Name:

Description: Micro Focus XDB ODBC 3

XDB Path: C:\Documents and Setting

Searching Filter: %

AuthID for ODBC tables: XDBODBC

Connection Options

Identifier Quoting: Default

DateTime Types: Date/Time

Connection

● Local ○ Remote

Server: LOCAL

Location: TUTORIAL

User ID:

Mode

● DB2 ○ XDB

Protocol

● NETBIOS ○ DRDA

○ SPX/IPX ○ TCP/IP

IpAddress:

TcpPort:

Optional Parameters

Current SQLID:

PackageSet:

□ With Hold Package

□ Reconnect

□ Read Only Queries

5. Enter the following details for the DSN (leave the remaining fields unchanged):

Data Source Name	GSDEMO

Server	DX40
User ID	install

6. Click **OK**. The DSN appears in the User Data Sources list.

See section *Run-time Knowledge Base Requirements* as you must create a Run-time Knowledge Base in order to mask or subset data.

Starting Data Builder

Data Builder enables functionality for Data Masking and Data Subset Extraction, and for the ODBC and Oracle Extension technologies. With Data Builder, you can take an inventory of data, and classify the data as you see fit. These processes store information about the data in the Data Express Knowledge Base.

Note: You will use the settings shown in this section throughout each subsequent chapter of this Getting Started guide.

Start Data Builder:

1. Click **Start** > **All Programs** > **Micro Focus Data Express 4.0** > **Data Builder**.
2. Select the XDB Knowledge Base Dist KB for DX 4.0 as the required database with which to launch the connection. By default, the highlighted item in the list of available databases is the last database you connected to using Data Builder. However, for the very first connection, Dist KB for DX 4.0 - DX40 is selected by default.

   Note: By selecting Dist KB for DX 4.0 - DX40, you do not need to choose the database and then the schema DX40.
3. Click **OK**.
4. In the **User Name** box, type: **install**.
5. In the **Password** box, type: **1**.

   Note: This is the number one.

   The**[Work with Data Stores]** window provides a list of files cataloged by Data Express using different sets of search criteria, including:

- Machine identifiers
- Company names
- Application identifiers
- Data store types
- Data stores with assigned classes (Multiple class Selection)
- Data stores with Data Changer activated
- Data stores with Data Changer inactivated
- Data stores set as Register Table
- Data stores without Register Table attribute

This window comprises four tabular views: Structure, Classes, Data Changer, and Register Tables. You can select a tab to view files using a specific set of criteria as listed previously. Each tab is associated with different grouping criteria.

The list of displayed data stores can be further restricted by applying filters. Filters can be selected from the context menu displayed by right-clicking the **List of Files** grid, or by selecting the specific icons displayed on the left side of the screen.

Note: Data Masking and the Data Subset Extraction are launched as separate applications.

Setting Options

Let's look at some ways you can configure Data Express behavior.

Default Environment Parameter Settings Selection

You can set default parameters for the environment, identified by the Machine ID and Company pair. The main parameter**, Output Date Format**, indicates the format in which dates are displayed in Data Express, for example:

- *DD/MM/YY* or *DD/MM/YYYY* - day/month/year (year with 2 or 4 digits)
- *MM/DD/YY* or *MM/DD/YYYY* - month/day/year (year with 2 or 4 digits)

Note: The **Output Date Format** parameter is valid for the whole environment and is therefore independent of the **Company** parameter. To define or change this parameter, click **Environment** > **Work With Default Parameters**.

Client Options

Options can be edited in the Options window:

1. To display the window, click **Tools** > **Options.**
2. In the **Help language** list, select **English**.
3. As a security measure to not store password information for the current connection with the Knowledge Base, leave the box **Save user name and password** unchecked.
4. Click the **Filter Settings** tab.

   The **Filter Settings** tab allows the scope of the filters configured in the **Work with Files** area to be selected.
5. Select **Filter on all TabSheets** so that all filtering properties remain regardless of what tab (**Structure**, **Classes**, **Data Changer**, or **Registers Tables**) in the **Data Builder - Work with data stores** window is active.
6. Leave the box **Show filter before loading data** unchecked.
7. Click the **Confirmation Messages** tab.

   The **Confirmation Messages** tab lets you set specific work modes.
8. Leave all the default settings, and click **Apply**.
9. Click **OK**.
10. If you're planning to go straight on to the next session, you can keep Data Builder open. Otherwise, either Click **File** > **Exit** or click the ☒button, and then click **Yes**.

Data Inventory

This tutorial describes the Data Inventory process, which includes preparing a work environment using Data Builder, and Definition and association of classes.

Note: Sessions in this tutorial should be done in order.

Work through the exercises in this tutorial only after you have completed all exercises in the previous tutorial.

To begin, you must already be logged in to Data Builder.

Sample Session

In this session, you:

- Set your workspace
- Load data store information into the Knowledge Base

Setting Your Workspace

Data Builder lets you set your workspace as part of the Data Inventory process. You must specify your workspace to organize and sort data on the basis of a logical model:

1. From Data Builder, open the **Work with Machine IDs** window (click or access it through **Environment** > **Work with Machine IDs**) to create a new Machine ID.
2. Click **New**.
3. Specify your name as the name of the machine ID in the **Machine ID** box. For this tutorial, type: **ALLIE**.
4. Type a description of your machine ID in the **Description** box. For this tutorial, type: **ALLIE'S ENVIRONMENT**.
5. Specify additional information if appropriate. For this tutorial, the additional fields have been left blank. Click **Apply**.
6. Click **OK** to save your specification and to close the window.
7. From Data Builder, open the **Work with Companies** window (click or access it through **Environment** > **Work with Companies**) to create a new company code:
8. Click **New**.
9. Select the **ALLIE** Machine ID you just created.
10. Specify **DXPROJ** as the name of your company code in the **Company Code** box.
11. In the **Company Description** box, type: **DX GETTING STARTED GUIDE**.
12. By default, the Life Cycle is activated. Leave this box selected.
13. By default, the **Is a Backup Company** checkbox is not selected. Leave this box unselected.

    During the definition of a company, the backup company associated with it can also be defined.

    A Backup Company is useful when all the information concerning the files that undergo the Life Cycle procedure needs to be preserved.

    Note: Only one version for each file is stored in the backup company.
14. Click **Apply**.

    The company you just created appears in the **List of Companies**.
15. Click **OK** to save your specification and to close the window.
16. Continue to the next section.

Loading your Data Store Information into the Knowledge Base

Perform this procedure to map and load the definitions from your data store into the XDB Knowledge Base. The information from your data store will be mapped to your workspace.

Load data store information into the Knowledge Base:

1. Open the **Distributed Loader** window by clicking .
2. Verify that the Machine ID **ALLIE** and the Company **DXPROJ** are selected.

    Note: If this is your first time using the Distributed Loader, **ALLIE** and **DXPROJ** will be automatically selected.
3. For this tutorial, leave the load option **Skip** selected.
4. In the **Source database** list, select **GSDEMO**.
5. In the **Schema** box, type: **SRC**.
6. In the **User name** box, type: **install**.
7. In the **Password** box, type: **1**.

    Note: This is the number one.
8. Click **Preview Tables** to view all the available tables in the source database.
9. Check the desired tables to load.

    Note: If you would like to limit the tables displayed by schema, in the **Schema** box, type the full schema name for the tables you want to view, and then click **Preview Tables** again.

By default, all tables are selected. You can deselect tables as appropriate.

10. Click **Load** to begin the export of the metadata into the Knowledge Base.
11. Click **Yes**. An **Insert tables** window appears, followed by an **Information** window. After the metadata has fully loaded, an **Information** window appears that displays data store information including the number of data stores.
12. Click **OK**.
13. Click **Yes** to display the log file in a text format.
14. Close the notepad view of the log file.
15. Click **Close** to close the Distributed Loader.

    The list of Data Stores in the **[Work with Data Stores]** window is updated automatically.
16. 16. If you're planning to go straight on to the next session, you can keep Data Builder open. Otherwise, either Click **File** > **Exit** or click the ☒button, and then click **Yes**.

Data Masking

Data Masking is used to manage the privacy of data contained in databases of applications that are either developed internally or purchased by external suppliers. The aim is to protect sensitive data such as name, surname, or company name according to privacy laws.

This tutorial explains the Data Masking configuration procedure, and describes the steps required to migrate the files in your application.

Note: Sessions in this tutorial should be done in order.

This tutorial walks you through the data classification and the data masking phases.

Begin this session only after you have set your workspace and loaded data store information into the Knowledge Base in tutorials *Using Data Express* and *Data Inventory*. You must already be logged in to Data Builder to begin.

Sample Session

In this session, you:

- Create a masking class
- Associate a masking class to data elements
- Start the Data Masking module
- Enable masking
- Export masking information
- Execute masking

Creating a Masking Class

You can create new classes to be used for masking purposes, or you can use a predefined class. In this exercise you will create a masking class named **ZIPCODE**.

1. In Data Builder, open the **Work with Classes** window (click or access it through **Environment** > **Work with Classes**).
2. Click **New**.
3. In the **Name** field, specify **ZIPCODE** as the name of the class you want to create.
4. In the **Full description** field, specify **ZIPCODE** as the description of the class.

    Note: The description will be visible as a label when you assign the class to data elements within data stores.
5. Specify **UDCZIP5** in the **Data Masking Routine** box.

Note: This masking routine is provided during product installation.

You can mask at the class level or at the data-element level. In this tutorial, we will be masking at the class level.

6. Select the Super Class **Privacy**.
7. Select the data type **Alpha**.
8. Leave the remainder of the fields at their default values. Click **Apply**.

   The class **ZIPCODE** is now listed at the bottom of **List of Classes**.
9. Click **OK**.
10. Continue to the next section.

Associating a Masking Class to Data Elements

In this exercise you will be manually associating the **ZIPCODE** class you just created with columns in tables **CUSTOMER** and **EMPLOYEE**.

1. Open the **Work with Data Elements** window (click ▤ or access it through **Environment** > **Work with Data Elements**).

   Note: If a message appears indicating that the default confirmation options have not been set, click **Yes** to set them; click **Apply**, and then **OK** in the **Options** window. However, this should not occur if you followed the previous steps in the tutorial *Using Data Express*.

   You can also access the **Work with Data Elements** window from the main window by selecting the file name, right-clicking to view the context menu, and then by clicking **Work with Data Elements of the Selected Data Store**.
2. Click the **Apply Filter** button to display all data stores:

   Note: To narrow down the list of data elements displayed, use the filter fields (**Machine ID**, **Company name**, and **Application ID**), and then click **Apply Filter**. To display all data elements, simply click **Apply Filter**.
3. Click the **Classes - Assignment** tab.
4. In the **All Classes** pane, expand the **Privacy** Super Class, and then select the class **ZIPCODE**.
5. Click the **Data Store** column to sort data elements by data store.
6. In the **List of Data Elements**, select the row for **CUSTOMER.ZIP**. (That is, select the row for the data store**CUSTOMER** and the name **ZIP**.)
7. Click the **Assign Class** button to assign the **ZIPCODE** class to **CUSTOMER.ZIP**.

   ZIPCODE now appears in the **Assigned Class** column for **CUSTOMER.ZIP**.
8. In the **List of Data Elements**, select the row for **EMPLOYEE.ZIP**. (That is, select the row for the data store **EMPLOYEE** and the name **ZIP**.)
9. Click the **Assign Class** button to assign the **ZIPCODE** class to **EMPLOYEE.ZIP**.

   ZIPCODE now appears in the **Assigned Class** column for **EMPLOYEE.ZIP**.
10. Click the **Close** button to close the **Work with Data Elements** window.
11. If you are planning to go straight on to the next tutorial *Simple Data Subset Extraction*, you can keep Data Builder open. Otherwise, either Click **File** > **Exit** or click the ☒ button, and then click **Yes**.
12. Continue to the next chapter.

Starting the Data Masking Module

To start Data Masking:

1. Click **Start** > **All Programs** > **Micro Focus Data Express 4.0** > **Data Masking**.
2. Click **OK**.

3. Select the XDB Knowledge Base Dist KB for DX 4.0 as the required database with which to launch the connection. By default, the highlighted item in the list of available databases is the last database you connected to using Data Masking.
4. Click OK.
5. In the User Name box, type: install.
6. In the Password box, type: 1.

   Note: This is the number one.
7. Click **OK**.
8. Select the schema DX40, and click OK.

   The Main Window comprises four main areas: Machine ID./Company grid, All Machine ID./Companies, Multiple Class Selection, and Object Distribution.

   The area containing the grid and the tabs Enabled and Disabled is referred to as the Machine ID./ Company area or the Grid area. The grid lists files related to the selected class and to the selected machine ID and company. The names of the forms, their meaning and the included information are outlined below.

   The **Enabled** tab shows a list of all active files (enabling a file means that it is visible in the export functionality in order to be prepared for masking), and all files, which have one or more specific classes related to their *data elements.*

   The **Object Distribution** area displays a pie chart and bar chart representation of the file types for the classes selected in the **Machine ID./Companies** or **Multiple Class Section** area. The pie chart shows the percentage distribution of file types, while the bar chart shows the numerical distribution of file types. For both charts, both enabled and disabled files are included. Click the minimize button (_) to minimize the **Object Distribution** area.
9. Continue to the next section.

Enabling Masking

Data Masking lets you to enable masking for individual data stores or all data stores, and to enable masking at the data-element level or class level.

- You can mask at the class level or at the data-element level. In this tutorial, we will be masking at the class level.

In this exercise you will activate masking for all data stores (**EMPLOYEE** and **CUSTOMER**) at the class (**ZIPCODE**) level.

Note: To mask at the data-element level, double-click the desired data store from the **Main Window**, right-click the desired data element, and then click **Data Changer Data Element Properties**. Click **Change**, click **Yes** to activate the Changer option, and finally click **Apply** to activate your setting.

Enable masking:

1. Click **Tools>Enable All Data Stores**.

   Note: Only the data stores that have assigned classes with masking routines defined will be enabled.
2. Click Yes to continue.

   The data stores are now listed in the table on the **Enabled** tab. For a list of data stores that do not have assigned classes, click the **Disabled** tab.
3. Continue to the next section.

Exporting Masking Information

In this exercise you will export your masked information so that the Extension Technology can mask your data.

Export masking information:

1. Click (Distributed Exporter) from the main window of Data Masking. The **Distributed Exporter** window is shown:
2. In the **Machine ID** list, click **ALLIE**.

   Note: If you have not created any additional Machine IDs, **ALLIE** will be selected by default.

3. In the **Company** list, click **DXPROJ**.

   Note: If you have not created any additional Company names, **DXPROJ** will be selected by default.

4. Click the **Select Target** button.

   Note: If you have already completed this tutorial once, your target database will already be listed. If this is the case, you just need to select the row for the **GSDEMO** database from the **Target Database List** and then click **OK**. You can then continue to step 9.

5. Click **Add**.
6. Enter the following details for the target (leave the remaining fields unchanged):

Database	GSDEMO
Type	ODBC
User name	install
Password	1

   Note: In this tutorial, the source and target databases are the same.

7. Click **OK** to close the **Add Target** window.

   The database you added now appears in the **Select Target Database** window.

8. Click **OK** to close the **Select Target Database** window.
9. In the **Distributed Exporter** window, in the **Target schema** field, specify **TGT**.

   Note: You must specify a **Target schema** name different from the **Source schema** name if the target database is the same as the source database.

10. In the **Output directory**, specify the **odbc\config** subfolder of your data directory. For more information, see the section *Data* of chapter *Extension Technology Directory* of *Part 3. Using Extension Technology*. For example: `c:\Documents and Settings\All Users\Application Data\Micro Focus\Data Express 4.0\odbc\config`

    Note: Files needed for the ODBC Extension and Oracle Extension must be placed in the **config** directory of your default data directory on Windows. In this tutorial you will be invoking the ODBC Extension from a Windows environment; therefore, you will specify this directory as your output directory.

11. Click **Export**.
12. Click **Yes**.
13. Click **OK**.
14. Click **Close**.
15. Exit Data Masking. Click **File** > **Exit** or click the ☒ button, and then click **Yes**.
16. Continue to the next section.

Executing Masking

By default, the Extension Technology will attempt to insert the processed data into the target database. Execute masking with Extension Technology:

1. From a MS-DOS prompt, change to the executables directory **c:\Program Files\Micro Focus\Data Express 4.0\odbc**.

2. 2. Type `dxestart` The target ODBC data store now contains masked tables **TGT.EMPLOYEE** and **TGT.CUSTOMER**.
3. The execution finished without errors, so you are prompted to view the log file. Type **Y** to view the file.
4. Close the text file.
5. Continue to the next section.

Verifying Results

Verify that the zip codes in the masked tables are in fact different:

1. Launch the SQL Wizard:

   ```
   ..\mfsql\bin\xwiz40n
   ```

2. Notice that the values in the **Zip Code** column for **SRC.EMPLOYEE** and **SRC.CUSTOMER** are different in **TGT.EMPLOYEE** and **TGT.CUSTOMER**.

Simple Data Subset Extraction

Data Subset Extraction is used to generate a reduced test environment.

In this tutorial, you perform basic data subset extraction.

Note: Sessions in this tutorial should be done in order.

This tutorial walks you through the data classification and the data subset extraction phases. This session involves creating simple subsetting classes.

Begin this session only after you have set your workspace and loaded data store information into the Knowledge Base in tutorials *Using Data Express* and *Data Inventory*. You must already be logged in to Data Builder to begin.

Sample Session

In this session, you:

- Create simple subsetting classes
- Associate simple subsetting classes to data elements
- Start the Data Subset Extraction module
- Create a group
- Create a simple method within that group
- Export simple subsetting method information
- Execute simple subsetting method

Creating Simple Subsetting Classes

You can create new classes to be used for subsetting purposes, or you can use a predefined class. In this exercise you will create subsetting classes named **DEPTNAM** and **DEPTNO**.

1. In Data Builder, open the **Work with Classes** window (click 🔔 or access it through **Environment** > **Work with Classes**).
2. Click **New**.
3. In the **Name** field, specify **DEPTNAM** as the name of the class you want to create.
4. In the **Full description** field, specify **DEPARTMENT NAME** as the description of the class.

   Note: The description will be visible as a label when you assign the class to data elements within files.

5. Select the Super Class **None**.

6. Select the data type **Alpha**.
7. Leave the remainder of the fields at their default values.
8. Click **Apply**. The class **DEPTNAM** appears at the end of **List of Classes**.
9. Click **New**.
10. In the **Name** field, specify **DEPTNO** as the name of the class you want to create.
11. In the **Full description** field, specify **DEPARTMENT NUMBER** as the description of the class.
12. Select the Super Class **None**.
13. Select the data type **Alpha**.
14. Click **Apply**. The class **DEPTNAM** appears at the end of **List of Classes**.
15. Click **OK** to close the **Work with Classes** window.
16. Continue to the next section.

Associating Simple Subsetting Classes to Data Elements

In this exercise you will be manually associating the classes **DEPTNAM** and **DEPTNO** you just created with columns in tables **DEPT** and **EMPLOYEE**:

1. Open the **Work with Data Elements** window (click ▤ or access it through **Environment** > **Work with Data Elements**).

   Note: If a message appears indicating that the default confirmation options have not been set, click **Yes** to set them; click **Apply**, and then **OK** in the **Options** window. However, this should not occur if you followed the steps in the tutorial *Using Data Express*.

   You can also access the **Work with Data Elements** window from the main window by selecting the file name, right-clicking to view the context menu, and then by clicking **Work with Data Elements of the Selected Data Store**.
2. Click the **Apply Filter** button to display all data stores:

   Note: To narrow down the list of data elements displayed, use the filter fields (**Machine ID**, **Company name**, and **Application ID**), and then click **Apply Filter**. To display all data elements, simply click **Apply Filter**.
3. Click the **Classes - Assignment** tab.
4. In the **All Classes** pane, expand the **None** Super Class, and then select the class **DEPARTMENT NUMBER**.
5. Click the **Data Store** column to data elements by data store.
6. In the **List of Data Elements**, select the row for **DEPT.DEPT**. (That is, select the row for the data store**DEPT** and the name **DEPT**.)

   Note: By selecting data elements, you are specifying to mask at the data-element level. If you do not select any data element, you will be masking at the class level.
7. Click the **Assign Class** button to assign the **DEPTNO** class to **DEPT.DEPT**.

   DEPARTMENT NUMBER now appears in the **Assigned Class** column for **DEPT.DEPT**.
8. In the **List of Data Elements**, select the row for **EMPLOYEE.DEPT**. (That is, select the row for the data store **EMPLOYEE** and the name **DEPT**.)
9. Click the **Assign Class** button to assign the **DEPTNO** class to **EMPLOYEE.DEPT**.

   DEPARTMENT NUMBER now appears in the **Assigned Class** column for **EMPLOYEE.DEPT**.
10. In the **All Classes** pane under the **None** Super Class, select the class **DEPARTMENT NAME**.
11. In the **List of Data Elements**, select the row for **DEPT.DNAME**. (That is, select the row for the data store**DEPT** and the name **DNAME**.)
12. Click the **Assign Class** button to assign the **DEPTNAM** class to **DEPT.DNAME**.

    DEPARTMENT NAME now appears in the **Assigned Class** column for **DEPT.DNAME**.
13. Click the **Close** button to close the **Work with Data Elements** window.

14. If you are planning to go straight on to the next tutorial *Data Subset Extraction using Referential Integrity*, you can keep Data Builder open. Otherwise, either click **File** > **Exit** or click the ☒ button, and then click **Yes**.
15. Continue to the next section.

Starting the Data Subset Extraction Module

To start Data Subset Extraction:

1. Click **Start > All Programs > Micro Focus Data Express 4.0 > Data Subset Extraction**.

   Note: If you are using Data Express on Windows Vista and User Access Control is enabled, you must run Data Subset Extraction as Administrator.
2. Select the XDB Knowledge Base Dist KB for DX for 4.0 as the required database with which to launch the connection .
3. Click **OK**.
4. In the User Name field, specify install.
5. In the Password field, specify 1.

   Note: This is a number one.
6. Click **OK**.
7. Select the schema DX40. The Data Subset Extraction window appears:
8. The Data Subset Extraction window comprises two main areas: the node pane and the grid area. The node pane contains a hierarchical structure that shows the grouping associated with each root-level node. The two root-level nodes are Groups and Creators.
9. Continue to the next section.

Creating a Group

The **Work with Groups** window lets you to define, view, modify, and delete groups.

Groups can be used to logically separate the methods created for each workspace. Each method must belong to a group.

To create a group:

1. Open the **Work with Groups** window (click or access it through **Environment > Work with Groups**).
2. Click **New**.
3. In the **Machine ID** list, click **ALLIE**.
4. In the **Company** list, click **DXPROJ**.
5. In the **Group name** box, specify **DX40GS**.
6. In the **Group description** box, type **DX 40 GETTING STARTED GUIDE**.
7. Click **Apply**.
8. Click **OK**.
9. Continue to the next section.

Creating a Simple Method

The New Method Wizard is used to create a new method, which is the extraction proposal containing the set of operations needed for the creation of the test environment. During method creation, all the information concerning the method is saved locally in a directory indicated by the user

Reduction can be obtained for data stores of a certain size and/or a particular type. In this case, you must be knowledgeable about the application to determine the appropriate selections. However, Data Subset Extraction can recognize and automatically create a first subset containing a series of files (called register table files) that are initially brought into the new test environment. Identifying register table files has the following advantages:

- You can create a method to extract only the register table file subset, which is normally valid for the creation of any test environment independently of the selection criteria applied. Once this method has been saved and confirmed, it can be run whenever necessary.

You can create a second file subset, which filters the register file subset you created, to produce a file subset that is reduced even further.

To create a simple method:

1. Launch the **New Method Wizard** (click or access it through **Environment > Create New Method**).
2. In the **Machine ID** list, click **ALLIE**.
3. In the **Company** list, click **DXPROJ**.
4. **Note:** Files belonging to different workspaces cannot be included.
5. Click **Next** to go to the next screen. **Note:** If a group has not previously been defined in the **Work with Groups** area, it can be defined from this screen, by simply specifying its name and description in a specific field.
6. In the **Group name** box, click **DX40GS**.
7. Text in the **Group description** box populates automatically.
8. In the **Method** box, type **SIMPLE**.
9. In the **Method description** box, type: **SIMPLE METHOD**.
10. Click **Next** to go to the next screen.
11. Select **Import all non-register table Data Stores** to include demographic files are registered in a dedicated elaboration step of the method.
12. Click to clear the **Import all register table Data Stores** box.
13. Click **Next** to go the next screen:
14. Select **DEPARTMENT NAME** to be used as the primary extraction criteria.
15. Click **Next**. Data Subset Extraction automatically extracts all the environment files containing the selected classes, entering them into a specific step. Once the method has been created, the remaining files (that is the files that are not sensitive to the selected classes) will initially be excluded from elaboration. During the Method Confirmation phase, they can be included in the method again. These files will then be integrally copied into the new environment generated by the method involved.
16. One or more classes related to each class selected as a primary extraction step can be indicated. The primary extraction criteria are listed in the **Selection class** field.
17. Select the class **DEPARTMENT NUM BER** from the **Other class** list.
18. Click **Add** to relate the class **DEPARTMENT NUMBER** to the class **DEPARTMENT NAME**. Data Subset Extraction automatically extracts all the environment files containing the related classes and inserts them in a further elaboration step.
19. Click **Finish**. The **Create Method** window is displayed.
20. Click Start.
21. Click OK.
22. Click **Confirm**. During method confirmation, specified filters are applied to the classes needed to perform the extraction, and the method information is imported from a temporary file to the Knowledge Base. This process, by default, will only include the data stores that contain these selected classes. If you would like to include your other data stores not related to the classes specified previously, check the option **Move excluded data stores into new step**. In this example, we will simply create a method based on the data stores that use classes **DEPTNAM** and **DEPTNO**.
23. Click **OK**.
24. In the **Choose a Filter Type** list, click **FILTER BY VALUE LIST**:
25. In one of the boxes on the lower-left corner, type **Sales**.

26. Click **OK** to close the **Set Filter to Selection Classes** window.
27. Click **OK** to confirm the method.
28. In the **Work with Method** window, click **Properties**.
29. Click **Active**.
30. Click **OK** to close the **Method Properties** window.
31. Click **OK** to close the **Work with Method** window.
32. Continue to the next section.

Exporting Simple Subsetting Method Information

Export simple subsetting method information.

1. Open the **Distributed Exporter** window by clicking from the main window:
2. Click **Select Target**.

   Note: If you have already completed the Masking tutorial, your target database will already be listed. If this is the case, you just need to select the row for the **GSDEMO** database from the **Target Database List** and then click **OK**; then proceed to step 7.
3. Click **Add**.
4. Enter the following details for the target (leave the remaining fields unchanged):

Database	GSDEMO
Type	ODBC
User name	install
Password	1

5. Click **OK**.
6. Click **OK**.
7. In the **Distributed Exporter** window, in the **Target schema** field, specify **TGT**.
8. In the **Output directory**, specify the **odbc\config** subfolder of your data directory. For more information, see the section *Data* of chapter *Extension Technology Directory* of *Part 3. Using Extension Technology*.
9. Click **Export**.
10. When prompted to continue elaboration, click **Yes**.
11. Click **OK**.
12. Click **Close** to close the **Distributed Exporter** window.
13. If you are planning to go straight on to the next tutorial *Data Subset Extraction using Referential Integrity*, you can keep Data Subset Extraction open. Otherwise, either Click **File** > **Exit** or click the ☒ button, and then click **Yes**.
14. Continue to the next section.

Executing Simple Subsetting Method

By default, the Extension Technology will attempt to insert the processed data into the target database. For testing purposes, it is possible to configure the Extension to instead write the processed data into text files as comma separated values (**.csv**).

Execute a simple subsetting method with Extension Technology:

1. From a MS-DOS prompt, change to the executables directory **c:\Program Files\Micro Focus\Data Express 4.0\odbc**.
2. 2. Type `dxestart`

   The target ODBC data store now contains reduced tables.
3. The execution finished without errors, so you are prompted to view the log file. Type **Y** to view the file.
4. Close the text file.

5. Continue to the next section.

Verifying Results

Verify that the tables are in fact different:

1. 1. Launch the SQL Wizard: ..\mfsql\bin\xwiz40n
2. Notice that the tables **TGT.EMPLOYEE** and **TGT.DEPT** have been reduced from **SRC.EMPLOYEE** and **SRC.DEPT**.

Data Subset Extraction using Referential Integrity

In this tutorial, you import class information using referential integrity.

Note: Sessions in this tutorial should be done in order.

This tuorial walks you through the data classification and the data subsetting phases. In this session, subsetting is based on referential integrity rules.

Begin this session only after you have completed the tutorial *Simple Data Subset Extraction*. You must already be logged in to Data Builder to begin.

Sample Session

In this session, you:

- Create referential integrity classes
- Create a referential integrity method
- Export referential integrity method information to Extension Technology

1. · Execute referential integrity method

Creating Referential Integrity Classes

In this exercise you will import referential integrity classes:

1. In Data Builder, open the **Work with Data Elements** window (click ▤ or access it through **Environment** > **Work with Data Elements**).

   Note: You can also access the **Work with Data Elements** window from the main window by selecting the file name, right-clicking to view the context menu, and then by clicking **Work with Data Elements of the Selected Data Store**.
2. From the **Select Data Elements** tab, click the **Apply Filter** button to display all data stores.

   Note: To narrow down the list of data elements displayed, use the filter fields (**Machine ID**, **Company name**, and **Application ID**), and then click **Apply Filter**. To display all data elements, simply click **Apply Filter**.
3. Click the **Classes - Assignment** tab.
4. Click **Import Class**.
5. Select the **Referential Integrity** import type:
6. In the **Machine ID** list, specify **ALLIE**.
7. In the **Company** list, specify **DXPROJ**.
8. In the **Database** list, specify **GSDEMO**.
9. Click **Show Schemas**.
10. Click **Import**.
11. Close the **Import Class** window.

    In the **All Classes** pane in the **Work with Data Elements** window, classes **CLA0201** and **CLA0202** now exist in the **None** Super Class.

12. In the **List of Data Store Data Elements**, notice:

- class **CLA0201** is assigned to **CUSTOMER.C_NO** and **ORDERS.C_NO**
- · class **CLA0202** is assigned to **ORDERS.O_NO** and **ITEMS.O_NO**

13. Close the **Work with Elements** window.
14. If you are planning to go straight on to the next tutorial *Data Subset Extraction using a Combined Data Element*, you can keep Data Builder open. Otherwise, either Click **File** > **Exit** or click the ✖ button, and then click **Yes**.
15. Continue to the next section.

Creating a Referential Integrity Method

Create a referential integrity method:

1. In Data Subset Extraction, open the **Import Method** window by clicking (Import Method from Referential Integrity).
2. Enter the following details for the method:

Machine ID	ALLIE
Company	DXPROJ
Group	DX40GS
Method (method name)	RI METHOD
Method text (method description)	RI METHOD
Class	CLA0201
Database	GSDEMO

3. Click **Import**.
4. Click **Properties**.
5. Click **Active** to activate the method.
6. Click **OK** to close the **Method Properties** window.
7. Click **OK** to close the **Work with Method** window.
8. Continue to the next section.

Exporting Referential Integrity Method Information to Extension Technology

Export your referential integrity method information.

1. Click (Distributed Exporter) from the main window of Data Subset Extraction.
2. Verify that the method **RI METHOD** is selected.
3. Click **Export**.
4. When prompted to continue elaboration, click **Yes**.
5. Click **OK**.
6. Click **Close** to close the **Distributed Exporter** window.
7. If you are planning to go straight on to the next tutorial *Data Subset Extraction using a Combined Data Element*, you can keep Data Subset Extraction open. Otherwise, either Click **File** > **Exit** or click the ✖ button, and then click **Yes**.
8. Continue to the next section.

Executing Referential Integrity Method

By default, the Extension Technology will attempt to insert the processed data into the target database. For testing purposes, it is possible to configure the Extension to instead write the processed data into text files as comma separated values (**.csv**).

Execute subset extraction with Extension Technology:

1. From a MS-DOS prompt, change to the executables directory **c:\Program Files\Micro Focus\Data Express 4.0\odbc**.
2. 2. Type `dxestart`

   The target ODBC data store now contains the table **TGT.CUSTOMER**, **TGT. ORDERS**, and **TGT.ITEMS**. The contents within these tables follow the referential integrity defined in the database:

   - Only records in the **TGT.ORDERS** table that have the same customer number as records in the **TGT.CUSTOMER** table are present.
   - Likewise, only records in the **TGT.ITEMS** table that have the same order number as records in the **TGT.ORDERS** table are present.
3. Continue to the next section.

Verifying Results

Verify that the tables are in fact different.

1. Launch the SQL Wizard:

   ```
   ..\mfsql\bin\xwiz40n
   ```
2. Notice that the tables **TGT.ORDERS** and **TGT.ITEMS** have been reduced.

Data Subset Extraction Using a Combined Data Element

In this tutorial, you change the selection criteria in the **RI METHOD** you created in the tutorial *Data Subset Extraction Using Referential Integrity* to use a *combined data element*. This is a more realistic portrayal of what you will do with RI methods - that is, supply initial criteria to the lead table to drive the referential integrity constraints throughout your method.

A combined data element is essentially a "dummy" data element that you create for ease of use when it comes to manipulating classes. Once a combined data element is created, you can perform operations against the data element like you would for other elements.

Note: Sessions in this tutorial should be done in order.

This tutorial walks you through the data classification and the data subsetting phases. In this session, subsetting is based on a combined data element.

Begin this session only after you have completed the tutorial *Data Subset Extraction Using Referential Integrity*. You must already be logged in to Data Builder to begin.

In this session, you:

- Create a combined data element
- Create a class on a combined data element
- Associate a class with a combined element
- Modify a referential integrity method to use a combined data element as a selection class
- Export referential integrity method to Extension Technology
- Execute referential integrity method

Creating a Combined Data Element

Create a combined data element:

1. 1.Launch Data Builder .
2. Double-click the row for **SRC.CUSTOMER**.

3. Use <Ctrl> to select the rows for the data element names STATE and BALANCE.
4. Right-click to display a context menu, and then click **Add to Combined Data Element**.
5. Change the Data element name to FLDSTBAL.
6. 5. In the Description box, specify ST BALANCE COMBINED FIELD.
7. Click Apply.
8. Click Close to close the Add to Combined Data Element window.

   The combined data element **FLDSTBAL** you just created is added to the **List of Data Elements** for the data store **SRC.CUSTOMER**.
9. Continue to the next section.

Creating a Class on a Combined Data Element

In this exercise, you will create the class STBAL on a combined data element.

1. In Data Builder, open the **Work with Classes** window (click 🔔 or access it through **Environment** > **Work with Classes**).
2. Click **New**.
3. In the **Name** field, specify **STBAL** as the name of the class you want to create.
4. In the **Full description** field, specify **STBAL** as the description of the class.

   Note: The description will be visible as a label when you assign the class to data elements within data stores.
5. Select the Super Class **None**.
6. 6. Select the data type **Alpha**.
7. 7. Leave the remainder of the fields at their default values.
8. 8. Click **Apply**.

   The class **STBAL** appears at the end of **List of Classes**.
9. 9. Click **OK** to close the **Work with Classes** window.

10.10. Continue to the next section.

Associating a Class with a Combined Data Element

In this exercise you will be manually associating the **STBAL** class you just created with columns in table **CUSTOMER**:

1. In Data Builder, open the **Work with Data Elements** window (click 📋or access it through **Environment** > **Work with Data Elements**).

   Note: You can also access the **Work with Data Elements** window from the main window by selecting the file name, right-clicking to view the context menu, and then by clicking **Work with Data Elements of the Selected Data Store**.
2. Click the **Apply Filter** button to display all data stores.

   Note: To narrow down the list of data elements displayed, use the filter fields (**Machine ID**, **Company name**, and **Application ID**) to narrow down the list of fields displayed, and then click **Apply Filter**. To display all fields, simply click **Apply Filter**.
3. Click the **Classes - Assignment** tab.
4. In the **All Classes** pane, expand the **None** Super Class, and then select the class **STBAL**.
5. Click the **Data Store Name** column to data elements by data store.
6. In the **List of Data Elements**, select the row for **CUSTOMER.FLDSTBAL**. (That is, select the row for the data store**CUSTOMER** and the name **FLDSTBAL**.)
7. Click the **Assign Class** button to assign the **STBAL** class to **CUSTOMER.FLDSTBAL**.

   STBAL now appears in the **Assigned Class** column for **CUSTOMER.FLDSTBAL**.

8. Click the **Close** button to close the **Work with Data Elements** window.
9. Exit Data Builder. (Click **File** > **Exit** or click the ☒ button, and then click **Yes**.)
10. Continue to the next section.

Modifying a Referential Integrity Method to Use a Combined Data Element as a Selection Class

Modify a referential integrity method.

1. Launch Data Subset Extraction.
2. In the **List of Methods in Group**, double-click the row for the method **RI METHOD**.
3. Maximize the **Work with Method** window.
4. In the **Data Stores Elaboration** list, right-click step **10**, point to **Elaboration**, and then click **Selection class / Filter properties**.
5. Select the row for the class STBAL.
6. Click the filter type FILTER BY RANGE:
7. Specify **MD** in the **From** and **To** boxes.
8. In the **Choose a data element** list, click **BALANCE**.
9. Specify **0** in the **From** box, and **1000** in the **To** box.
10. Click **OK** to close the **Selection class / Filter properties** window.
11. Click **OK** to close the **Work with Method** window.
12. Continue to the next section.

Exporting a Referential Integrity Method to Extension Technology

Export method information:

1. Open the **Distributed Exporter** window by clicking from the main window.
2. Verify that all values are populated as shown previously.
3. Click **Export**.
4. When prompted to continue elaboration, click **Yes**.
5. Click **OK** to close the **Distributed Exporter** window.
6. Exit Data Subset Extraction. (Click **File** > **Exit** or click the ☒ button, and then click **Yes**.)
7. Continue to the next section.

Executing Referential Integrity Method

Execute subset extraction with Extension Technology:

1. From a MS-DOS prompt, change to the executables directory **c:\Program Files\Micro Focus\Data Express 4.0\odbc**.
2. Type `dxestart`

   The target ODBC data store now contains the table **TGT.CUSTOMER**, **TGT. ORDERS**, and **TGT.ITEMS**. The contents within these tables follow the referential integrity defined in the database:

   - Records in the **TGT.CUSTOMER** table have been reduced to only contain MD records with a balance of less than $1000 (USD).
   - Records in the **TGT.ORDERS** table have been reduced to only contain records that have the same customer number as the parent table **TGT.CUSTOMER**.
   - Likewise, records in the **TGT.ITEMS tables** have the same order number as records in the parent **TGT.ORDERS table**
3. Use standard tools and procedures relevant to your data store to verify that the information in the reduced tables is in fact different.

Verifying Results

Verify that the tables are in fact different:

1. Launch the SQL Wizard:

   ```
   ..\mfsql\bin\xwiz40n
   ```

2. Notice that the tables **TGT.ORDERS**, **TGT.CUSTOMER**, and **TGT.ITEMS** have been reduced.

Simple Sampling

In this tutorial, you perform sampling to obtain information about the distribution of data. Then you use that distribution as a way to assign classes to other data elements with similar data distributions.

Known Restriction: Data Express does not support sampling for binary data.

Note: Sessions in this tutorial should be done in order.

This tutorial walks you through the data preparation, data inventory, and data sampling phases.

Begin this session only after you configured connectivity for the sample XDB data store provided in the product installation in the tutorial *Using Data Express*. We recommend that you begin this session only after you have completed all previous tutorials.

You must already be logged in to Data Builder to begin.

Sample Session

The sample sessions are broken out into three parts: one to add the demo sampling tables to your XDB database TUTORIAL location, one to perform data inventory, and one to perform the actual sampling.

Note: Sessions in this tutorial must be done in order.

Database Preparation for Sampling Exercises

In this session, you prepare your sampling data by adding tables to your XDB database TUTORIAL location, which is necessary to complete the sampling exercises.

Note: The tables needed for this tutorial are different than the ones required for the previous tutorials.

1. From a MS-DOS prompt, type:

   ```
   cd
            InstallationDirectory\mfsql\bin
   ```

   where `InstallationDirectory` specifies the directory where Data Express 4.0 is installed, for example, **c:\Program Files\Micro Focus\Data Express 4.0**.
2. Load the data by executing the following command:

   ```
   xwiz40n /b sampling.sql
   ```

3. Once the command has finished running, continue to the next section.

Data Inventory

In this session, you prepare your Data Express environment for the sampling data stores and add metadata for the sampling data stores into the Knowledge Base:

- Set your sampling workspace
- Load data store information into the Knowledge Base

Setting Your Sampling Workspace

Data Builder lets you set your workspace as part of the Data Inventory process. You must specify your workspace to organize and sort data on the basis of a logical model:

1. From Data Builder, open the **Work with Companies** window (click or access it through **Environment** > **Work with Companies**) to create a new company code.

   Note: Notice that the Machine ID**ALLIE** is listed with the **DXPROJ** Company Code in the **List of Companies**. You created **ALLIE** during the tutorial *Data Inventory*. For the purpose of this tutorial, you create a new company under the **ALLIE** Machine ID. If for some reason you did not perform the *Data Inventory* tutorial, you must create a Machine ID prior to performing this step.
2. Click **New**.
3. Select the **ALLIE** Machine ID.
4. Specify **SAMPLING** as the name of your company

   code in the **Company Code** box.
5. In the **Company Description** box, type: **SAMPLING EXERCISE**.
6. By default, the Life Cycle is activated. Leave this box selected.
7. By default, the **Is a Backup Company** checkbox is not selected. Leave this box unselected.

   During the definition of a company, the backup company associated with it can also be defined.

   A Backup Company is useful when all the information concerning the files that undergo the Life Cycle procedure needs to be preserved.

   Note: Only one version for each file is stored in the backup company.
8. Click **Apply**. The company you just created appears in the **List of Companies**.
9. Click **OK** to save your specification and to close the window.
10. Continue to the next section.

Loading Data Store Information into the Knowledge Base

Perform this procedure to map and load the definitions from your data stores into the XDB Knowledge Base. The information from your data stores will be mapped to your workspace.

Load data store information into the Knowledge Base:

1. Open the **Distributed Loader** window by clicking .
2. Select the Machine ID **ALLIE** and the Company **SAMPLING**.
3. For this tutorial, leave the load option **Skip** selected.
4. In the **Source database** list, select **GSDEMO**.
5. In the **Schema** box, type: **SAMPLING**.
6. In the **User name** box, type: **install**.
7. In the **Password** box, type: **1**.

   Note: This is the number one.
8. Click **Preview Tables** to view all the available tables in the source database.
9. Click **Load** to begin the export of the metadata into the Knowledge Base.
10. Click **Yes**.

    An **Insert tables** window appears, followed by an **Information** window. After the metadata has fully loaded, an **Information** window appears that displays data store information including the number of data stores.
11. Click **OK**.
12. 12. Click **Yes** to display the log file in a text format.

13.13. Close the notepad view of the log file.

14.14. Click **Close** to close the Distributed Loader. The list of data stores in the **[Work with Data Stores]** window is updated automatically.

15.15. Keep Data Builder open, and continue to the next section.

Data Element Sampling

The data for this session is actually located in an XDB database that resides on the Windows machine where you are running your Data Express client software. Typically when this is the case, you would simply execute sampling by clicking **Start** in the **Distributed Sampler** window; there is no need to export sampling configuration, execute sampling, and import results.

In this session however, sampling is executed in a similar fashion as it would be in a UNIX environment. Therefore, sampling configuration must be exported, sampling executed, and results imported.

In this session, you:

- Set and export sampling configuration
- Execute sampling
- Import sampling results
- Verify results
- Associate a fingerprint to a class
- Associate a class to data elements

Setting and Exporting Sampling Configuration

Perform this procedure to verify your sampling configuration settings and to export information about the data store to output files.

Note: This exercise contains several references to the **config** subdirectory. For more information, see chapter *Extension Technology Directories*.

1. In the **List of Data Stores** in the **Work with Data Stores** window, select a row for a **SAMPLING.*TableName*** data store.
2. Right-click and select **Enable Sampling**.

   Enabling sampling changes the *classification number* value for the data store to a **1**.

   Classification numbers are used to restrict the data stores that get sampled. By default, all data stores are assigned a **Classification number** of **0**, which means that sampling is disabled. When sampling the data stores within your work environment, all enabled data stores with **Classification number** values of **1** will be sampled.
3. Repeat steps 1-2 for each **SAMPLING.*TableName*** data store.
4. In the **List of Data Stores**, select a data store with the Machine ID **ALLIE** and the company **SAMPLING**, right-click, and then click **Data Store Properties**:
5. Note that the **Classification number** value is a **1**.

   Classification numbers can be changed in the **Properties** window to provide further granularity in restricting which data stores get sampled as based on user-defined criteria. The data store remains enabled as long as the **Classification number** is not **0**.However, just because sampling is enabled does not mean that the data store is a candidate for sampling as this is controlled by **Classification number** in the **Distributed Sampler**.

   If the **Classification number** set for the data store is less than or equal to the **Classification number** specified in the **Distributed Sampler**, the data store will be sampled. Likewise, if the data store **Classification number** is greater than the **Classification number** in the **Distributed Sampler**, the data store will not be sampled.

   Tip: You can assign classification numbers to your data stores based on how often you want to sample data. For example, **1** could represent daily sampling, **2** could represent **weekly** sampling, and **3** could represent monthly sampling.

6. Open the **Distributed Sampler** window by clicking .
7. Select the **ALLIE** Machine ID if it is not already selected.
8. Select the **SAMPLING** company code.
9. Retain all default sampling options.

   For this exercise, all data stores should be sampling candidates and all sampling options should be utilized.

   The following list describes the sampling **Options**:

   - **Compressed sampling** - Produces the data element *fingerprint*. The fingerprint graphically shows the distribution of values within a given range for the sampled data element. The fingerprints for numeric and alphanumeric data elements differ in that the fingerprint for an alphanumeric data element shows the distribution of values based on the first character and provides additional information.
   - **Standard sampling -** Displays information for each data element value including the number of times each value occurred and the percentage that value represents in the total population.
   - **Min/Max calculation** - Displays the minimum and maximum values for the data element.
10. Note the **Output directory** specified. By default, this is the directory path to your **config** subdirectory, which is located in the **data** directory appropriate to the ODBC Extension.

    Notes:

    - By default, the directory path to the **config** subdirectory is listed for the ODBC Extension. If you are using the Oracle Extension, this must be changed.
    - You can change the **Output directory** to any directory; just make sure that the files **sampling.dat** and **method.rc** are copied to the appropriate **config** directory prior to executing sampling.
11. Click **Advanced**.
12. Retain all default values.

    Note: The **Data Element Size** of **0** (zero) means to include all sizes in the sampling results and does not actually reflect the value of 0 when **Ignore Special Values Zero / Space** is checked. Likewise, the default **0** values in **Max. And Min. Recalculation Additional Options** mean to consider all values to be in-range and does not reflect the value of 0.
13. Click **OK** to return to the **Distributed Sampler** window.
14. Click **Export**:
15. Click **OK**.

    Two output files have been created and are located in the output directory on your Windows machine:

    - **sampling.dat -** contains information regarding the data elements to be sampled
    - **method.rc -** contains encrypted connectivity information about the source data store (data store name, username, and password)
16. Verify that the files **sampling.dat** and **method.rc** are located in the **config** subdirectory.

    Important: Under normal circumstances, you would need to copy the files **sampling.dat** and **method.rc** to the **config** subdirectory on the machine where your source data store is located. However, in this exercise the files are already in the correct location.
17. Continue to the next section.

Executing Sampling

During sampling execution, the Extension Technology uses the output files **sampling.dat** and **method.rc** to produce the **sampling.log** file that contains the resulting statistics.

1. From a MS-DOS prompt, change to the Extension Technology executables directory. For example, change to **c:\Program Files\Micro Focus\Data Express 4.0\ODBC**.

Note: For more information about directories, see *Extension Technology Directories*.

2. Type `dxesampling`.
3. Change to the log directory.

   Note: For more information, see section *Executables* in *Extension Technology Directories*.
4. Verify that the **sampling.log** file is listed.
5. Continue to the next section.

Loading Sampling Results

The **sampling.log** file that was generated during sampling execution must be loaded into the Knowledge Base.

1. Open the **Distributed Sampler** window by clicking .
2. Verify that the Machine ID selected is **ALLIE**.
3. In the **Company** list, click **SAMPLING**.
4. Click the **Load results from file** tab.
5. Click and browse for the **sampling.log** file that was generated during sampling execution.
6. Click **Load**.
7. Continue to the next section

Verifying Results

Once you have exported sampling configuration information, executed sampling, and loaded the results into the Knowledge Base, you can view the results.

When compressed sampling is performed, a *fingerprint* is created. The fingerprint is a unique graphical representation of the distribution of data for the sampled data element. When the results for compressed sampling are loaded, the fingerprint is created.

1. In Data Builder, click the **Sampling** tab:
2. Select the row for the **SAMPLING.TABADD** data store name in the **List of data stores - Sampling Active** list.
3. Click (**Show Synthetic Data Elements**) or right-click and then select **Show Synthetic Data Element Contents**.

   The **Data Store Data Elements** list in the left pane shows the data elements for the data store you selected.
4. In the **Data Store Data Elements** list, select the data element **PRGADDR**.

   You can also select the image for the data element in the **Data Element Samples** grid. To ensure that you have selected the correct one, the related data element name is highlighted in the **Data Store Data Elements** list.

   Sampling results for the numeric data element **PRGADDR** are displayed.
5. Review the data element fingerprint in the **Zoomed Data Element Sample**. The fingerprint is the result of performing compressed sampling.

   Notice the following:

   - The numbers on the vertical y-axes represent the ranges of values based on the decimal place, where each number represents one decimal place to the left of the decimal point. For instance, 0 represents the ones place (0-9), 1 represents the tens place (10-99), 2 represents the hundreds place (100-999), and so on.
   - The numbers on the horizontal x-axes represent a specific range based on the corresponding y-coordinate. For instance, **1** represents the value 1 for the y-coordinate **0**, while **1** represents values 10-19 for the y-coordinate **1**.

- The values in each cell indicate the number of times a value in the data element falls within the specified range.

Note: The fingerprint can also be viewed as a bar graph by clicking **Graph**.

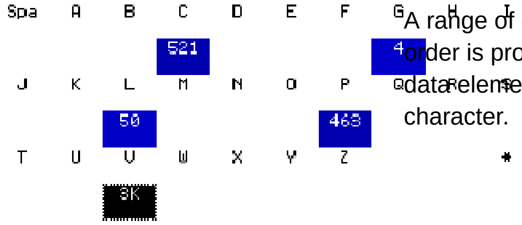
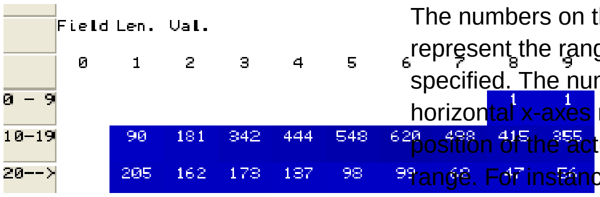
6. Review the items in the list **Sample Analysis of the Selected Data Element**. This list shows the actual data element values when standard sampling was performed.

   Known Restriction: Currently Data Express only shows the first 1000 distinct values and ranges, which are sorted either numerically or alphanumerically based on the data element type. When there are more than 1000 distinct data element values, the **Data Element Value** ends at the 1000th unique value instead of the greatest data element value.
7. In the **Data Store Data Elements** list, select the data element **ADDRESS**.

   Sampling results for alphanumeric data element **ADDRESS** are displayed.

   Review the data element fingerprint in the **Zoomed Data Element Sample**. The fingerprint is the result of performing compressed sampling.

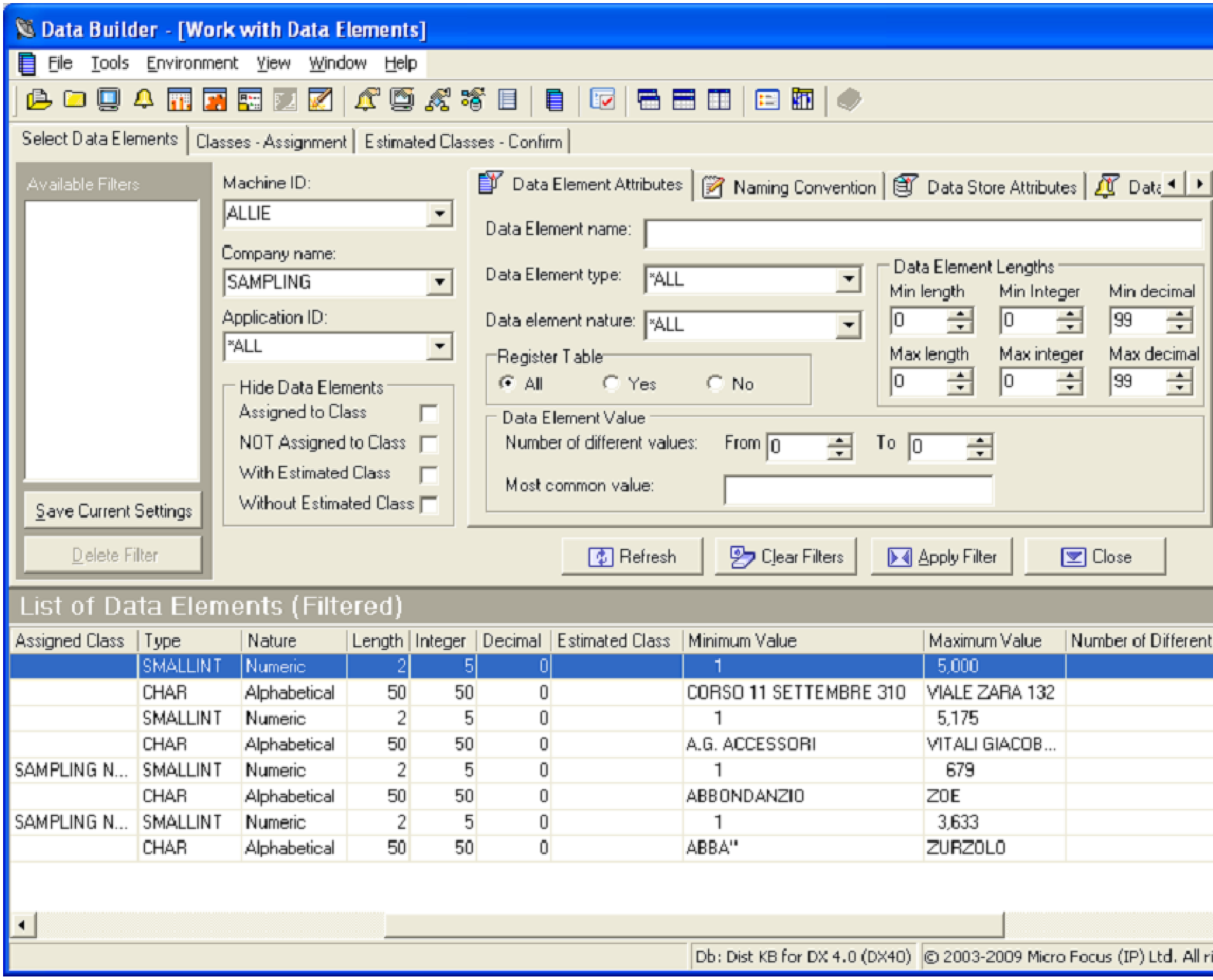
   Notice that the fingerprint for an alphanumeric data element contains four distinct sections:

Section Type	Example	Description
Character Distribution	 Spa A B C D E F G H I 521 4 J K L M N O P Q R S 58 463 T U V W X Y Z * 9K	A range of characters in alphabetical order is provided to show how many data elements begin with that character.
Number Distribution		A range of numbers is provided to show how many data elements begin with the indicated number. In the provided example, no data elements begin with numbers.
Type Summary	Num. Only Alpha Only 76	An alphanumeric data element can actually be a numeric-only data element or alphanumeric data element. The type summary provides a count of how many data elements fall into either category. In the provided example, all data elements are alphanumeric in type.
Field Length	 Field Len. Val. 0 1 2 3 4 5 6 7 8 9 0 - 9 1 1 10-19 90 181 342 444 548 620 498 415 355 20--> 205 162 173 137 98 99 68 47 56	The numbers on the vertical y-axes represent the ranges of values specified. The numbers on the horizontal x-axes represent the position of the actual number in that range. For instance, **90** in the provided example means that there are 90 values with a length equal to 11.

Note: The fingerprint can also be viewed as a bar graph by clicking **Graph**.

8. Close the **Show Synthetic Data Element Contents** window to return to **Work with Data Stores**.

9. Open the **Work with Data Elements** window (click or access it through **Environment** > **Work with Data Elements**).
10. In the **Machine ID** list, click **ALLIE**.
11. In the **Company name** list, click **SAMPLING**.
12. Click **Apply Filter**.



Machine ID: ALLIE

Company name: SAMPLING

Application ID: *ALL

Hide Data Elements
- Assigned to Class □
- NOT Assigned to Class □
- With Estimated Class □
- Without Estimated Class □

Data Element name:

Data Element type: *ALL

Data element nature: *ALL

Register Table: ● All ○ Yes ○ No

Data Element Lengths: Min length 0, Min Integer 0, Min decimal 99, Max length 0, Max integer 0, Max decimal 99

Data Element Value: Number of different values: From 0 To 0

Most common value:

List of Data Elements (Filtered)

Assigned Class	Type	Nature	Length	Integer	Decimal	Estimated Class	Minimum Value	Maximum Value	Number of Different
	SMALLINT	Numeric	2	5	0		1	5,000	
	CHAR	Alphabetical	50	50	0		CORSO 11 SETTEMBRE 310	VIALE ZARA 132	
	SMALLINT	Numeric	2	5	0		1	5,175	
	CHAR	Alphabetical	50	50	0		A.G. ACCESSORI	VITALI GIACOB...	
SAMPLING N...	SMALLINT	Numeric	2	5	0		1	679	
	CHAR	Alphabetical	50	50	0		ABBONDANZIO	ZOE	
SAMPLING N...	SMALLINT	Numeric	2	5	0		1	3,633	
	CHAR	Alphabetical	50	50	0		ABBA"	ZURZOLO	

Db: Dist KB for DX 4.0 (DX40) © 2003-2009 Micro Focus (IP) Ltd. All ri

13. Note the **Minimum Value** and **Maximum Value** items in the **List of Data Elements**, which are the result of performing min/max calculation sampling.
14. Leave the **Work with Data Elements** window open, and continue to the next section.

Creating a Sampling Class

You can create a new class to be used for sampling purposes, or you can use a predefined class. In this exercise you will create a sampling class named **SAMPNUM**.

1. Open the **Work with Classes** window (click or access it through **Environment** > **Work with Classes**).
2. Click **New**.
3. In the **Name** box, specify **SAMPNUM** as the name of the class you want to create.
4. In the **Full description** field, specify **SAMPLING NUMERIC VALUES** as the description of the class.

Note: The description will be visible as a label when you assign the class to data elements within data stores.

5. Select the Super Class **Numbers**.
6. Select the data type **Numeric**.
7. Leave the remainder of the fields at their default values. Click **Apply**.

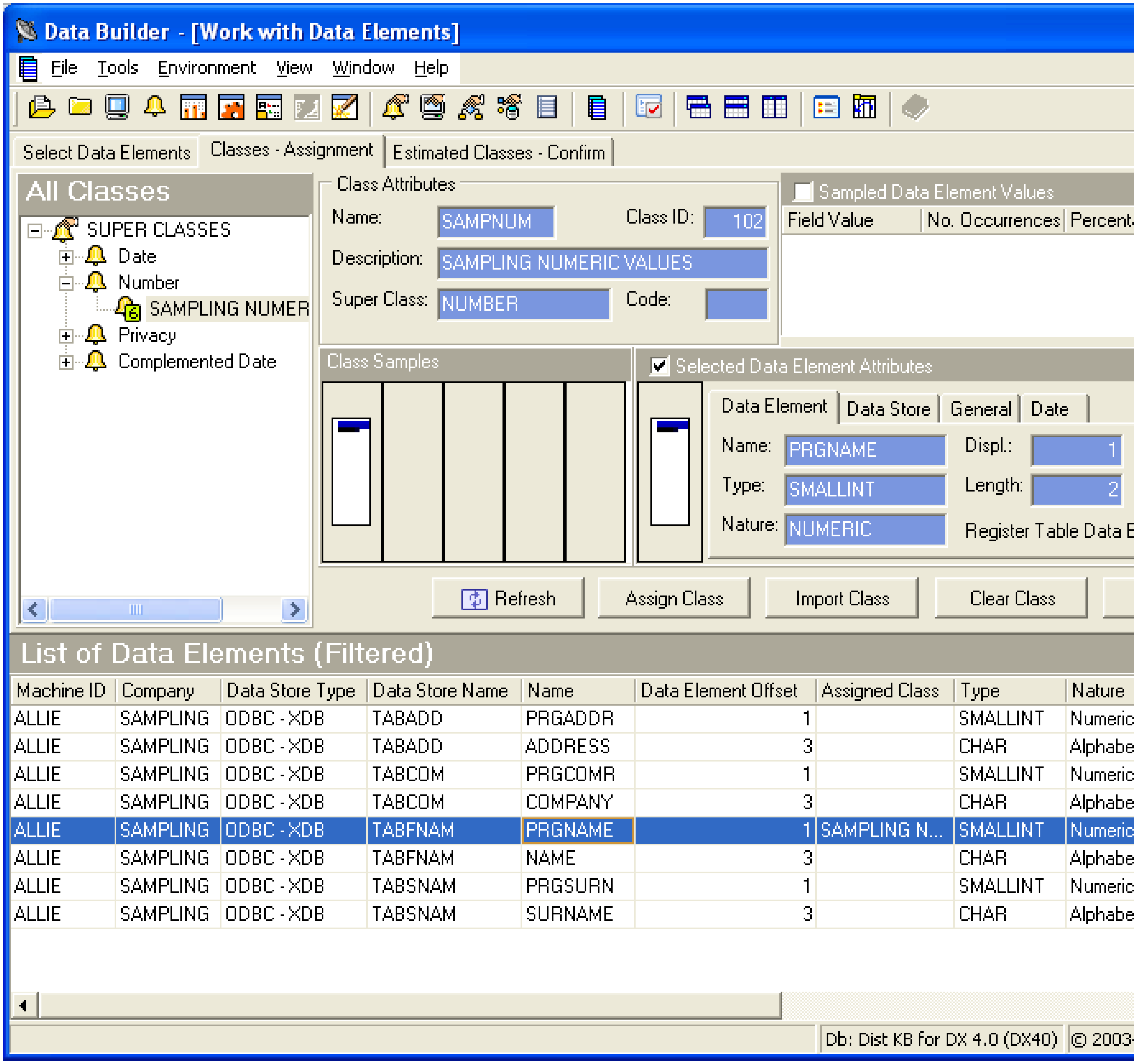
   The class **SAMPNUM** is now listed in the **List of Classes**.
8. Click **OK** to return to the **Work with Data Elements** window.
9. Click **Refresh**.
10. Continue to the next section.

Associating a Fingerprint to a Class

In Data Express, you can correlate data elements if their value distributions are similar. To do this, you must first associate a fingerprint, which represents the desired distribution, to a class.

This fingerprint becomes the *prototype* that is used determine the class assignment for other data elements. If the prototype and the fingerprint for a data element are deemed similar based on a calculated confidence value, the data element is also associated with the prototype class.

1. Click the **Classes - Assignments** tab.
2. In the **All Classes** pane, expand the **Number** Super Class, and select the class description **SAMPLING NUMERIC VALUES**.
3. In the **List of Data Elements**, select the row for the data element **PRGNAME** in the data store **TABFNAM**.
4. Check the **Selected Data Element Attributes** check box:
5. Drag the fingerprint that now appears beneath the text **Selected Data Element Attributes**, to the first **Class Samples** cell.



Class Attributes

Name: SAMPNUM Class ID: 102

Description: SAMPLING NUMERIC VALUES

Super Class: NUMBER Code:

□ Sampled Data Element Values

Field Value	No. Occurrences	Percent

☑ Selected Data Element Attributes

Name: PRGNAME Displ.: 1

Type: SMALLINT Length: 2

Nature: NUMERIC

List of Data Elements (Filtered)

Machine ID	Company	Data Store Type	Data Store Name	Name	Data Element Offset	Assigned Class	Type	Nature
ALLIE	SAMPLING	ODBC - XDB	TABADD	PRGADDR	1		SMALLINT	Numeric
ALLIE	SAMPLING	ODBC - XDB	TABADD	ADDRESS	3		CHAR	Alphabe
ALLIE	SAMPLING	ODBC - XDB	TABCOM	PRGCOMR	1		SMALLINT	Numeric
ALLIE	SAMPLING	ODBC - XDB	TABCOM	COMPANY	3		CHAR	Alphabe
ALLIE	SAMPLING	ODBC - XDB	TABFNAM	PRGNAME	1	SAMPLING N...	SMALLINT	Numeric
ALLIE	SAMPLING	ODBC - XDB	TABFNAM	NAME	3		CHAR	Alphabe
ALLIE	SAMPLING	ODBC - XDB	TABSNAM	PRGSURN	1		SMALLINT	Numeric
ALLIE	SAMPLING	ODBC - XDB	TABSNAM	SURNAME	3		CHAR	Alphabe

By highlighting the **SAMPNUM** class description in the **All Classes** pane and then by dragging the fingerprint for the **PRGNAME** data element to the **Class Samples** cell, the **SAMPNUM** class is then assigned to the fingerprint. This fingerprint becomes the prototype. This action also associates the **SAMPNUM** class to the **PRGNAME** data element.

6. Continue to the next section.

Associating a Class to Data Elements

In this exercise you are associating the class **SAMPNUM** (which is associated to your sampling prototype) to other data elements with sampling data distributions similar to the prototype fingerprint.

The level of similarity between distributions depends on the thresholds you set when importing the class information. If the prototype and the fingerprint for a data element are deemed similar based on a calculated confidence value, the data element is also associated with the prototype class.

1. From the **Work with Data Elements** window, click **Import Class**. The **Import Class** window is displayed.

2. Click **Sampling Results**.

   When comparing the fingerprint for a data element to the prototype fingerprint, internal Data Express confidence values are calculated to provide measurements that illustrates the similarities between the data distributions. The internal confidence values are used as input for the threshold formulas.
3. In the **Machine ID** list, click **ALLIE**.
4. In the **Company** list, click **SAMPLING**.
5. In the **Super Classes** section, click **Numbers**.
6. Set the **Theshold 1** value to **82**.

   The value for **Threshold 1** represents a percentage where the similar areas in the two fingerprints are weighted more heavily than the dissimilar areas.
7. Keep the logical operator set to **AND**.
8. Set the **Theshold 2** value to**75**.

   The value for **Threshold 2** represents a percentage where all areas (both similar and dissimilar) in the two fingerprints are given equal weight.
9. Click **Import**:
10. Click **OK**.
11. Close the **Import Class** window and return to the **Work with Data Elements** window:
12. Click the **Close** button to close the **Work with Data Elements** window.

    A sampling class has successfully been assigned to data elements with similar distributions to the prototype.
13. Close Data Builder; either Click **File** > **Exit** or click the ☒ button, and then click **Yes**.

Data Generation

In this tutorial, you perform basic data generation, working with the TGT.DEPT and TGT.EMPLOYEE tables to populate a table based on the content of another table and on data in a dictionary.

Begin this session only after you have set your workspace and loaded data store information into the Knowledge Base by following the instructions in tutorials *Using Data Express* and *Data Inventory*. You must be logged into Data Builder before you begin.

In this session, you:

- Catalog the XDB tables provided in the sample
- Start the Data Generator module.
- Create simple Data Generation code
- Submit JCL to generate data

Complete these tutorials in the following order:

Catalog and Classify Tables

1. Using a different machine and company, repeat the cataloguing phase, but loading only the TGT.DEPT and TGT.EMPLOYEE tables.
2. Execute the classification phase as described in the *Data Subset Extraction* part of this guide, assigning the same class to the DEPTNO columns of both TGT.DEPT and TGT.EMPLOYEE.

Start the Data Generation Module

1. Start the **Data Generation** module from your **Start** menu or tile, depending on your Windows operating system.

   Note: If you are using Data Express on Windows Vista and User Access Control is enabled, you must run the Data Generation module as Administrator.

2. Select the **XDB Knowledge Base** as the required database for your ODBC connection.

   By default, the highlighted item in the list of available databases is the last database you connected to using Data Masking. This becomes apparent after you click **Available databases**.
3. Click **OK**.
4. Log in using your username and password.
5. Click **OK**.
6. Select the schema **DX40**; click **OK**.

Create Simple Data Generation Code

1. Launch the New Code Wizard from the **New Code** icon.
2. Specify the proper machine and company.
3. Give the code the name `SIMPLEC` and provide a description.
4. Click **Apply**.

   This creates the code.
5. Select the code, and look at the lower part of the screen; it shows the laws that are used in the code, one law per table.
6. Ensure that the **TGT.EMPLOYEE** table appears before the **TGT.DEPT** table. If now, reorder the tables using the arrows on the right.
7. Right-click on **TGT.EMPLOYEE**; then select **Set directive** from the context menu.
8. Select the **DEPT** column; then right-click and select **Assign output class** specifying the **DEPT** column.
9. Right-click **TGT.EMPLOYEE**; then select **Don’t create output** on the context menu.
10. Right-click **TGT.DEPT**; then select **Set directive** from the context menu.
11. Right-click the COD_CUS column; then select **Set directive** from the context menu, specifying **Other file dependencies** as the directive type.
12. Right-click **TGT.EMPLOYEE**; then select **Set directive** on the context menu.
13. Select the **DNAME** column; then select **Set directive** from the context menu, specifying **Dictionary** as the directive type and **USDCHSUR** as dictionary name.
14. Export the information about the tables DEPT and EMPLOYEE using the Distributed Exporter.

Execute a Simple Data Generation Law

By default, Extension Technology attempts to insert the processed data into the target database.

Execute a simple data generation law with Extension Technology:

1. From a command prompt, change to the executables directory, which by default is `%PROGRAMFILES%\Micro Focus\Data Express 4.0\Synthetic\ODBC`.
2. Enter `dxestart`.

   This populates the tables in the target ODBC data store using the specified data generation laws. After the execution finishes without errors, you are prompted to view the log file.
3. Type Y to view the file.
4. Close the text file.
5. Continue to the next section.

Verify Results

Verify that the tables are populated according to definition:

1. Launch the SQL Wizard: `..\mfsql\bin\xwiz40n`
2. Verify that the **TGT.EMPLOYEE** table contains its original values, and the **TGT.DEPT** table has been populated according to the specified laws.

Oracle Alias Creation

In order to use the Oracle Extension, you must first create an alias in the Borland Database Engine (BDE) for your source data store to communicate with the Oracle client installed on your machine. A BDE alias is not required for your target data store.

Note:

- For your source Oracle data store specified for either the Distributed Loader or Distributed Exporter, you must create a BDE alias. We recommend that the associated BDE alias match the Oracle TNS service name specified in your **tnsnames.ora** file.
- For your target Oracle data store specified in the Distributed Exporter, make sure that the target name is the Oracle TNS service name specified in your **tnsnames.ora** file located on the machine where you intend to mask/subset data.
- If you are using Data Express on Windows Vista and User Access Control is enabled, you must run the BDE Administrator utility as **Administrator**.

To create an Oracle alias:

1. Open BDE Administrator (from the **Control Panel**). Click the **Configuration** tab, and then select **ORACLE** from the **Native Drivers** list as shown in *Figure A-1:*
2. In the **Definition of ORACLE** pane, click in the second column for **DLL32** to display a list of DLLs. Select **SQLORA8.DLL**, and click **Object** > **Apply** as shown in *Figure A-2*:
3. 3. In the **Definition of ORACLE** pane, click in the second column for **VENDOR INIT**, and enter **OCI.DLL** as the Vendor DLL file name.
4. 4. In the **Drivers and System** pane, click the **Databases** tab.
5. 5. Create a new Oracle database alias (right-click, select **New**, select the Database Driver Name **ORACLE**, and then **OK**) and rename the alias as appropriate.
6. 6. In the **Definition of *AliasName*** pane, click in the second column for **SERVER NAME**, and enter your Oracle TNS.

Oracle Wallet

Oracel Wallet allows the Data Express engine to connect the source and target databases without needing to specify the user name and password.

Oracle Wallet is a container/repository that stores credentials such as certificates, certificate requests, and private keys. You can use this feature if you enable HTTPS. Oracle Wallets can be stored either on the file system or an LDAP Server (like OID).

You manage Oracle Wallets using Oracle Wallet Manager (OWM - java based application). OWM is executable in `$ORACLE_HOME/bin` .

Using OWM you can:

- Create wallets
- Create certificate requests
- Import certificates to wallets
- Upload wallets to a directory

Implementing Oracle Wallet Support

There is a new parameter in the `dxeconfig.cfg` file named `Identify Externally` that identifies the connection criteria. This parameter is at the end of the file in the section applicable to the Oracle Extension only.

The possible values are `Y` and `N`. Default value is `N`.

Yes Data Express will skip the reading of user name and password from method.rc. In this case, the connection to the database, instead of happening in the usual way (using the user and password specified during the “export” phase), will use the configured Wallet (if the Wallet has been configured properly). The system will connect to the database without needing explicit user name and password credentials.

No Data Express behavior will remain unchanged and access continues to require a valid user name and password.

The user interface will remain unchanged (and in case of use of wallet feature, the data specified as user and password are ignored), and the only change needed, from the user perspective, is that in order to use the new feature, you must set the `dxeconfig.cfg` configuration file properly.

Here is a sample of the `dxeconfig.cfg` file:

```
* Data Express Configuration File

LogLevel:1
Always Drop Table:N
CSV Data Store:N
COBOL Masking:Y
Validate Target Table:Y
Validate Source Schema:N
Enable Logging:N
Hide Progress:N

* These entries are only applicable to the Oracle Extension.

Text Output:N
Data Separator:,
Fields Enclosed By:"
Identify Externally:Y
```

The `dxeconfig` file is not used for sampling. In order to use wallet management, you need to specify a `–e` parameter.

Use the following command to use identification via user and password: `Dxestart`

Use the following command to use identification via wallet: `Dxestart -e`

Index